V850E2/MN4 マイクロコンピュータ

USB CDC（コミュニケーション・デバイス・クラス）ドライバ編

R01AN0010JJ0101
Rev.1.01
2011.04.28

要旨

このアプリケーション・ノートは，V850E2/MN4 マイクロコントローラに内蔵の USB ファンクション・コントローラ用に作成された CDC（コミュニケーション・デバイス・クラス）用サンプル・ドライバについて説明します。

主に次に示す内容で構成されます。

- サンプル・ドライバの仕様
- サンプル・ドライバを利用したアプリケーション・プログラム開発のための環境
- サンプル・ドライバを利用するための参考情報

動作確認デバイス

V850E2/MN4(uPD70F3512)搭載 RTE-V850E2/MN4-EB-S

目次

1. はじめに

1.1 注　意

このアプリケーション・ノートで使用するサンプル・プログラムはあくまで参考用のものであり，当社がこの動作を保証するものではありません。
サンプル・プログラムを使用する場合，ユーザのセット上で十分な評価をしたうえで使用してください。

1.2 対象者

このアプリケーション・ノートは，V850E2/MN4 マイクロコントローラの機能を理解し，それを用いたアプリケーション・システムを開発しようとするユーザを対象とします。

1.3 目　的

このアプリケーション・ノートは，V850E2/MN4 マイクロコントローラに内蔵の USB ファンクション・コントローラを使用するためのサンプル・ドライバの仕様をユーザに理解していただくことを目的とします。

1.4 構　成

このアプリケーション・ノートは，大きく分けて次の内容で構成しています。

- USB 規格の概要
- サンプル・ドライバの仕様
- 開発環境(CubeSuite/Multi(注 1)/ IAR Embedded Workbench (注 2))
- サンプル・ドライバの応用

(注 1) Multi は Green Hills SoftwareTM, Inc.の登録商標です。
(注 2) IAR Embedded Workbench は IAR システムズ株式会社の登録商標です。

1.5 読み方

このマニュアルの読者には，電気，論理回路，およびマイクロコントローラに関する一般知識を必要とします。

— V850E2/MN4 マイクロコントローラのハードウエア機能，および電気的特性を知りたいとき
　→ 別冊の V850E2/MN4 マイクロコントローラのユーザーズマニュアル ハードウエア編を参照してください。

— V850E2/MN4 マイクロコントローラの命令機能を知りたいとき
　→ 別冊の V850E2M ユーザーズマニュアル アーキテクチャ編を参照してください。

2. 概　説

このアプリケーション・ノートは，V850E2/MN4 マイクロコントローラに内蔵の USB ファンクション・コントローラ用に作成された CDC（コミュニケーション・デバイス・クラス）用サンプル・ドライバについて説明します。主に次に示す内容で構成されます。

- サンプル・ドライバの仕様
- サンプル・ドライバを利用したアプリケーション・プログラム開発のための環境
- サンプル・ドライバを利用するための参考情報

この章では，サンプル・ドライバの概要と，適用対象となるマイクロコントローラについて説明します。

2.1 概　要

2.1.1 USBファンクション・コントローラの特徴

サンプル・ドライバの制御の対象である V850E2/MN4 マイクロコントローラの USB ファンクション・コントローラには，次のような特徴があります。

- USB（Universal Serial Bus Specification）2.0 に準拠
- フル・スピード（12 Mbps）デバイスとして動作
- エンドポイントを次のように構成

表 2-1　V850E2/MN4 マイクロコントローラのエンドポイント構成

エンドポイント名	FIFO サイズ（バイト）	転送タイプ	備考
Endpoint0 Read	64	コントロール転送（IN）	－
Endpoint0 Write	64	コントロール転送（OUT）	－
Endpoint1	64×2	バルク転送 1（IN）	2 バッファ構成
Endpoint2	64×2	バルク転送 1（OUT）	2 バッファ構成
Endpoint3	64×2	バルク転送 2（IN）	2 バッファ構成
Endpoint4	64×2	バルク転送 2（OUT）	2 バッファ構成
Endpoint7	64	インタラプト転送（IN）	－
Endpoint8	64	インタラプト転送(IN)	－

- USB 標準リクエストには自動応答（一部のリクエストを除く）
- バス・パワードとセルフ・パワードを選択可能
- 内部クロックと外部クロックを選択可能 (注 3)

  内部クロック：外部 9.6MHz×内部 20 逓倍÷4 分周（48 MHz）
  　　または外部 7.2 MHz×内部 20 逓倍÷3 分周（48 MHz）

  外部クロック：USBCLK 端子へ入力（fUSB = 48 MHz）

(注 3) サンプル・ドライバでは内部クロックを選択します。

2.1.2　サンプル・ドライバの特徴

V850E2/MN4 マイクロコントローラ向け CDC（コミュニケーション・デバイス・クラス）用サンプル・ドライバには，次のような特徴があります。機能や動作の詳細は第 4 章　サンプル・ドライバの仕様を参照してください。

- USB コミュニケーション・デバイス・クラス Ver.1.1 の Abstract Control Model に準拠
- 仮想 COM デバイスとして動作
- 次に示すサイズのメモリを占有（ベクタ・テーブルを除く）
  ROM：約 4.6 K バイト
  RAM：約 0.8 K バイト

2.1.3　サンプル・ドライバの構成

サンプル・ドライバは CubeSuite 版・Multi 版・IAR Embedded Workbench 版の 3 種類が用意されています。開発環境に合わせてご使用ください。
各サンプル・ドライバは次のようなファイルで構成されています。

（1）CubeSuite 版

CubeSuite 版サンプル・ドライバのファイル構成は，以下のようになっています。

表 2-2　CubeSuite 版サンプル・ドライバのファイル構成

フォルダ	ファイル	概要
src	main.c	メイン・ルーチン，初期化，サンプル・アプリケーション
	usbf850.c	USB 初期化，エンドポイント制御，バルク転送，コントロール転送
	usbf850_communication.c	CDC 固有処理
	cstart.asm	ブートストラップ
include	main.h	main.c 関数プロトタイプ宣言
	usbf850.h	usbf850.c 関数プロトタイプ宣言
	usbf850_communication.h	usbf850_communication.c 関数プロトタイプ宣言
	usbstrg_desc.h	ディスクリプタ定義
	usbf850_errno.h	エラー・コード定義
	usbf850_types.h	ユーザ型宣言
	reg_v850e2mn4.h	USB ファンクション用レジスタ定義
Inf ファイル	XXX_CDC_VISTA.inf	Windows®Vista®用 INF ファイル ****の部分は各マイコン名が入ります： MN4
	XXX_CDC_WIN7.inf	Windows7®用 INF ファイル ****の部分は各マイコン名が入ります： MN4
	XXX_CDC_XP.inf	Windows XP®用 INF ファイル ****の部分は各マイコン名が入ります： MN4

備考　この他，CubeSuite（ルネサスエレクトロニクス製統合開発ツール）用プロジェクト関連ファイル一式も同梱されています。詳細は「6. 2. 1　ホスト環境整備」を参照してください。

（2）Multi 版

Multi 版サンプル・ドライバのファイル構成は，以下のようになっています。

表 2-3　Multi 版サンプル・ドライバのファイル構成

フォルダ	ファイル	概要
src	main.c	メイン・ルーチン，初期化，サンプル・アプリケーション
	usbf850.c	USB 初期化，エンドポイント制御，バルク転送，コントロール転送
	usbf850_communication.c	CDC 固有処理
	initial.s	ブートストラップ
	vector.s	割り込みベクタテーブル宣言
include	main.h	main.c 関数プロトタイプ宣言
	usbf850.h	usbf850.c 関数プロトタイプ宣言
	usbf850_communication.h	usbf850_communication.c 関数プロトタイプ宣言
	usbstrg_desc.h	ディスクリプタ定義
	usbf850_errno.h	エラー・コード定義
	usbf850_types.h	ユーザ型宣言
	reg_v850e2mn4.h	USB ファンクション用レジスタ定義
	df3512_800.h	V850E2/MN4 用レジスタ定義
Inf ファイル	XXX_CDC_VISTA.inf	Windows®Vista®用 INF ファイル ****の部分は各マイコン名が入ります： MN4
	XXX_CDC_WIN7.inf	Windows7®用 INF ファイル ****の部分は各マイコン名が入ります： MN4
	XXX_CDC_XP.inf	Windows XP®用 INF ファイル ****の部分は各マイコン名が入ります： MN4

備考　この他，Multi（Green Hills SoftwareTM, Inc.製統合開発ツール）用プロジェクト関連ファイル一式も同梱されています。詳細は「6. 4. 1 ホスト環境整備」を参照してください。

（3）IAR Embedded Workbench 版

IAR Embedded Workbench 版サンプル・ドライバのファイル構成は，以下のようになっています。

表 2-4　IAR Embedded Workbench 版サンプル・ドライバのファイル構成

フォルダ	ファイル	概要
src	main.c	メイン・ルーチン，初期化，サンプル・アプリケーション
	usbf850.c	USB 初期化，エンドポイント制御，バルク転送，コントロール転送
	usbf850_communication.c	CDC 固有処理
include	main.h	main.c 関数プロトタイプ宣言
	usbf850.h	usbf850.c 関数プロトタイプ宣言
	usbf850_communication.h	usbf850_communication.c 関数プロトタイプ宣言
	usbstrg_desc.h	ディスクリプタ定義
	usbf850_errno.h	エラー・コード定義
	usbf850_types.h	ユーザ型宣言
	reg_v850e2mn4.h	USB ファンクション用レジスタ定義
Inf ファイル	XXX_CDC_VISTA.inf	Windows®Vista®用 INF ファイル ****の部分は各マイコン名が入ります： MN4
	XXX_CDC_WIN7.inf	Windows7®用 INF ファイル ****の部分は各マイコン名が入ります： MN4
	XXX_CDC_XP.inf	Windows XP®用 INF ファイル ****の部分は各マイコン名が入ります： MN4

備考　この他，IAR Embedded Workbench 用プロジェクト関連ファイル一式も同梱されています。詳細は「6. 6. 1 ホスト環境整備」を参照してください。

2.2 V850E2/MN4 マイクロコントローラ

サンプル・ドライバの制御の対象である V850E2/MN4 マイクロコントローラの詳細については，該当する製品のユーザーズマニュアル　ハードウエア編を参照してください。

2.2.1 適用製品

サンプル・ドライバは，次に示す製品に適用できます。

表 2-5　V850E2/MN4 マイクロコントローラ製品一覧

愛　称	品　名	内蔵メモリ		内蔵 USB 機能	割り込み		UM
		フラッシュ・メモリ	RAM		内部 注 4	外部 注 4	
V850E2/MN4	μPD70F3510	1MB	64 KB ＋64KB	Host / Function	180	29	V850E2/MN4 ユーザーズマニュアル ハードウエア編 (R01UH0011JJ0002)
	μPD70F3512	1MB	64 KB ＋64KB	Host / Function	190	29	
	μPD70F3514	1MB	64 KB × 2 ＋64KB	Host / Function	196	29	
	μPD70F3515	2MB	64 KB × 2 ＋64KB	Host / Function	196	29	

(注 4)　ノンマスカブル割り込みを含みます。

2.2.2 特　徴

V850E2/MN4 には，主に次のような特徴があります。

- 内蔵メモリ
  RAM： シングル・コア 64K バイト　デュアル・コア 64K バイト×2
  フラッシュ・メモリ： 1 M バイト

- フラッシュ・キャッシュ
  シングル・コア：　16K バイト（4 ウェイ・セット・アソシアティブ）
  デュアル・コア：　16K バイト（4 ウェイ・セット・アソシアティブ）×2

- 外部バス・インタフェース
  2 系統のメモリ・コントローラを搭載
  プライマリ・メモリ・コントローラ（SRAM/SDRAM 接続可能）
  セカンダリ・メモリ・コントローラ（SRAM/SDRAM 接続可能）

- シリアル・インタフェース
  アシンクロナス・シリアル・インタフェース UART : 6ch
  クロック同期式シリアル・インタフェース CSI : 6ch
  アシンクロナス・シリアル・インタフェース UART(FIFO) : 4ch
  クロック同期式シリアル・インタフェース CSI(FIFO) : 4ch
  I2C : 6 チャネル
  CAN : 2 チャネル（μ PD70F3512,70F3514,70F3515）
  USB ファンクション・コントローラ : 1 チャネル
  USB ホスト・コントローラ : 1 チャネル
  イーサネット・コントローラ : 1 チャネル（μ PD70F3512, 70F3514, 70F3515）

- DMA コントローラ
  DMA コントローラ : 16 チャネル
  DTS : 最大 128 チャネル

3. USBの概要

この章では，サンプル・ドライバが準拠する USB 規格の概要を説明します。
USB（Universal Serial Bus）は共通のコネクタでさまざまな周辺機器をホスト・コンピュータに接続できるようにするためのインタフェース規格です。ハブと呼ばれる分岐点を追加することで最大 127 個の機器を接続でき，Plug&Play で機器を認識できるホットプラグに対応しているなど，従来のインタフェースより柔軟で使いやすくなっています。現在では PC の USB インタフェース搭載率はほぼ 100%になってきており，PC と周辺機器間の標準インタフェースとして定着したと言えます。
USB 規格の策定と管理は USB Implementers Forum（USB-IF）という団体が行っています。USB 規格の詳細は USB-IF の公式ウェブサイト（www.usb.org）を参照してください。

3.1 転送方式

USB 規格では，4 種類の転送方式（コントロール，バルク，インタラプト，アイソクロナス）が定義されています。各転送方式には表 3－1 に示す特徴があります。

表 3-1　USB の転送方式

項目＼転送方式		コントロール転送	バルク転送	インタラプト転送	アイソクロナス転送
特徴		周辺機器の制御などに必要な情報のやりとりに使用される転送方式	非周期的に大量データを扱う転送方式	周期的でバンド幅が低いデータ転送方式	リアルタイム性が要求される転送方式
設定可能なパケット・サイズ	ハイ・スピード 480 Mbps	64 バイト	512 バイト	1-1024 バイト	1-1024 バイト
	フル・スピード 12 Mbps	8, 16, 32, 64 バイト	8, 16, 32, 64 バイト	1-64 バイト	1-1023 バイト
	ロウ・スピード 1.5 Mbps	8 バイト	－	1-8 バイト	－
転送の優先順位		3	3	2	1

3.2 エンドポイント

エンドポイントはホスト・デバイスが通信相手を特定するための情報の 1 つで，0-15 の番号と方向（IN/OUT）で指定されます。エンドポイントは周辺機器で使用するデータ通信経路ごとに用意しなければならず，複数の通信経路で共用できません(注 5)。たとえば，SD カードへの書き込み／読み出しとプリント出力の機能を持った機器の場合，SD カードへの書き込み用エンドポイント，読み出し用エンドポイント，プリント出力用エンドポイントを個別に持つ必要があります。どのような機器でも必ず使用するコントロール転送には，エンドポイント 0 を使用します。

データ通信を行うとき，ホスト・デバイスは機器を特定する USB デバイス・アドレスとともにエンドポイント（番号と方向）を使用して，機器内部の通信先を特定します。

エンドポイントのための物理的な回路として周辺機器内にバッファ・メモリを装備し，USB と通信先（メモリなど）の速度差を吸収する FIFO の役割も果たします。

(注 5) オルタナティブ設定という仕組みを使い，排他的に切り替える方法はあります。

3.3 クラス

USB を介して接続する周辺機器（ファンクション・デバイス）には，その機能によりさまざまなデバイス・クラスが定義されています。代表的なクラスとしてマス・ストレージ・クラス（MSC），コミュニケーション・デバイス・クラス（CDC），ヒューマン・インタフェース・デバイス・クラス（HID）などがあります。各デバイス・クラスにはプロトコルなどで標準仕様が定められているため，これに準拠していれば共通のホスト・ドライバを使用できます。

コミュニケーション・デバイス・クラス（CDC）は，ホスト・コンピュータに接続する通信機器のためのクラスで，モデム，FAX，ネットワーク・カードなどが対象となります。最近では PC に RS-232C インタフェースが搭載されなくなっていることから，PC と UART 通信を行う際の USB シリアル変換を実現するデバイスに使われることが多くなっています。なお，CDC には実装する機器によっていくつかのモデルが定義されています。サンプル・ドライバはこのなかの Abstract Control Model を使用しています。

3.4　リクエスト

USB 規格では，ホスト・デバイスからすべてのファンクション・デバイスに対してリクエストと呼ばれるコマンドを発行することにより通信が開始されます。リクエストには処理の方向，種類，ファンクション・デバイスのアドレスなどのデータが含まれています。各ファンクション・デバイスはリクエストをデコードして自身に対するリクエストかを判定し，自身に対するリクエストの場合にだけ応答します。

3.4.1　種　類

標準リクエスト，クラス・リクエスト，ベンダ・リクエストの 3 種類があります。
サンプル・ドライバが対応するリクエストについては，「4. 1. 2　リクエストへの対応」を参照してください。

（1）標準リクエスト

すべての USB 対応機器で共通に使用するリクエストです。bmRequestType フィールドのビット 6，5 の値がともに 0 のとき，そのリクエストは標準リクエストです。各標準リクエストの処理内容については，USB 仕様書（Universal Serial Bus Specification Rev.2.0）を参照してください。

表 3-2　標準リクエスト一覧

リクエスト名	対象ディスクリプタ	概要
GET_STATUS	デバイス	電源（セルフ／バス）とリモート・ウエイクアップの設定の読み取り
	エンドポイント	Halt 状態の読み取り
CLEAR_FEATURE	デバイス	リモート・ウエイクアップのクリア
	エンドポイント	Halt の解除（DATA PID = 0）
SET_FEATURE	デバイス	リモート・ウエイクアップまたはテスト・モードの設定
	エンドポイント	Halt の設定
GET_DESCRIPTOR	デバイス，コンフィギュレーション，ストリング	対象ディスクリプタの読み取り
SET_DESCRIPTOR	デバイス，コンフィギュレーション，ストリング	対象ディスクリプタの変更（オプション）
GET_CONFIGURATION	デバイス	現行設定のコンフィギュレーション値の読み取り
SET_CONFIGURATION	デバイス	コンフィギュレーション値の設定
GET_INTERFACE	インタフェース	対象インタフェースの現行設定のうちオルタナティブ設定値の読み取り
SET_INTERFACE	インタフェース	対象インタフェースのオルタナティブ設定値の設定
SET_ADDRESS	デバイス	USB アドレスの設定
SYNCH_FRAME	エンドポイント	フレーム同期のデータ読み取り

(2) クラス・リクエスト

デバイス・クラス固有のリクエストです。サンプル・ドライバでは CDC の Abstract Control Model に対応したクラス・リクエストへの応答処理を実装しています。応答可能なリクエストは次のとおりです。

- Send Encapsulated Command
  コミュニケーション・クラス・インタフェースの制御プロトコルのフォーマットでコマンドを発行するためのリクエストです。
- Get Encapsulated Response
  コミュニケーション・クラス・インタフェースの制御プロトコルのフォーマットで応答を要求するためのリクエストです。
- Set Line Coding
  シリアル通信の通信フォーマットを指定するためのリクエストです。
- Get Line Coding
  デバイス側の現在の通信フォーマット設定を取得するためのリクエストです。
- Set Control Line State
  RS-232/V.24 形式の制御信号のためのリクエストです。

(3) ベンダ・リクエスト

ベンダ・リクエストは，ベンダが独自に定義するリクエストです。ベンダ・リクエストを使用する場合，ベンダはそのリクエストに対応するホスト・ドライバを提供する必要があります。
bmRequestType フィールドのビット 6 の値が 1，ビット 5 の値が 0 のとき，そのリクエストはベンダ・リクエストです。

3.4.2 フォーマット

USB リクエストは 8 バイト長で，次のようなフィールドで構成されています。

表 3-3　USB リクエストのフォーマット

オフセット	フィールド		説明
0	bmRequestType		リクエストの属性
		ビット 7	データ転送方向
		ビット 6, 5	リクエスト・タイプ
		ビット 4-0	対象ディスクリプタ
1	bRequest		リクエスト・コード
2	wValue	下位	リクエストで使用する任意の数値
3		上位	
4	wIndex	下位	リクエストで使用するインデックスまたはオフセット
5		上位	
6	wLength	下位	データ・ステージでの転送バイト数（データ長）
7		上位	

3.5 ディスクリプタ

USB 規格では，各ファンクション・デバイス固有の情報を定められた形式でコード化したものをディスクリプタと呼んでいます。ファンクション・デバイスは，ホスト・デバイスからのリクエストに応じてディスクリプタを送信します。

3.5.1 種　類

次に示す 5 種類のディスクリプタが定義されています。

- デバイス・ディスクリプタ
  どのデバイスにも必ず存在するディスクリプタで，対応している USB 仕様のバージョン，デバイス・クラス，プロトコル，Endpoint0 に対する転送で利用可能な最大パケット長，ベンダ ID，プロダクト ID などの基本情報が含まれています。
  GET_DESCRIPTOR_Device リクエストに応答して送信するディスクリプタです。

- コンフィギュレーション・ディスクリプタ
  すべてのデバイスに 1 つ以上存在するディスクリプタで，デバイスの属性（電源供給方法），消費電力などの情報を含みます。
  GET_DESCRIPTOR_Configuration リクエストに応答して送信するディスクリプタです。

- インタフェース・ディスクリプタ
  インタフェースごとに必要なディスクリプタで，インタフェース識別番号，インタフェース・クラス，サポートするエンドポイントの数などが含まれます。
  GET_DESCRIPTOR_Configuration リクエストに応答して送信するディスクリプタです。

- エンドポイント・ディスクリプタ
  インタフェース・ディスクリプタに指定されたエンドポイントごとに必要なディスクリプタで，転送タイプ（転送方向），転送で利用可能な最大パケット長，転送のインターバルを定義します。ただし，Endpoint0 はこのディスクリプタを持ちません。
  GET_DESCRIPTOR_Configuration リクエストに応答して送信するディスクリプタです。

- ストリング・ディスクリプタ
  任意の文字列を含むディスクリプタです。
  GET_DESCRIPTOR_String リクエストに応答して送信するディスクリプタです。

3.5.2 フォーマット

ディスクリプタのサイズとフィールドは，次のように種類ごとに異なります。各フィールドのデータ並びはリトル･エンディアンです。

表 3-4　デバイス・ディスクリプタのフォーマット

フィールド	サイズ（バイト）	説　明
bLength	1	ディスクリプタのサイズ
bDescriptorType	1	ディスクリプタの種類
bcdUSB	2	USB 仕様リリース番号
bDeviceClass	1	クラス・コード
bDeviceSubClass	1	サブクラス・コード
bDeviceProtocol	1	プロトコル・コード
bMaxPacketSize0	1	Endpoint0 の最大パケット・サイズ
idVendor	2	ベンダ ID
idProduct	2	プロダクト ID
bcdDevice	2	デバイスのリリース番号
iManufacturer	1	製造者を表すストリング・ディスクリプタへのインデックス
iProduct	1	製品を表すストリング・ディスクリプタへのインデックス
iSerialNumber	1	デバイスの製造番号を表すストリング・ディスクリプタへのインデックス
bNumConfigurations	1	コンフィギュレーションの数

備考　　ベンダ ID：USB デバイスを開発する各企業が USB-IF から取得する識別番号
　　プロダクト ID：ベンダ ID を取得後，各企業が自社製品に割り振る識別番号

表 3-5　コンフィギュレーション・ディスクリプタのフォーマット

フィールド	サイズ（バイト）	説　明
bLength	1	ディスクリプタのサイズ
bDescriptorType	1	ディスクリプタの種類
wTotalLength	2	コンフィギュレーション，インタフェース，およびエンドポイント・ディスクリプタの総バイト数
bNumInterfaces	1	このコンフィギュレーションが持つインタフェースの数
bConfigurationValue	1	このコンフィギュレーションの識別番号
iConfiguration	1	このコンフィギュレーションを記述するストリング・ディスクリプタへのインデックス
bmAttributes	1	このコンフィギュレーションの特徴
bMaxPower	1	このコンフィギュレーションの最大消費電流（2 □A 単位）

表 3-6　インタフェース・ディスクリプタのフォーマット

フィールド	サイズ（バイト）	説　　明
bLength	1	ディスクリプタのサイズ
bDescriptorType	1	ディスクリプタの種類
bInterfaceNumber	1	このインタフェースの識別番号
bAlternateSetting	1	このインタフェースに対するオルタナティブ設定の有無
bNumEndpoints	1	このインタフェースが持つエンドポイントの数
bInterfaceClass	1	クラス・コード
bInterfaceSubClass	1	サブクラス・コード
bInterfaceProtocol	1	プロトコル・コード
iInterface	1	このインタフェースを記述するストリング・ディスクリプタへのインデックス

表 3-7　エンドポイント・ディスクリプタのフォーマット

フィールド	サイズ（バイト）	説　　明
bLength	1	ディスクリプタのサイズ
bDescriptorType	1	ディスクリプタの種類
bEndpointAddress	1	このエンドポイントの転送方向 このエンドポイントのアドレス
bmAttributes	1	このエンドポイントの転送タイプ
wMaxPacketSize	2	この転送の最大パケット・サイズ
bInterval	1	このエンドポイントのポーリング間隔

表 3-8　ストリング・ディスクリプタのフォーマット

フィールド	サイズ（バイト）	説　　明
bLength	1	ディスクリプタのサイズ
bDescriptorType	1	ディスクリプタの種類
bString	任意	任意のデータ列

4. サンプル・ドライバの仕様

この章では，V850E2/MN4 マイクロコントローラ向け USB コミュニケーション・デバイス・クラス（CDC）用サンプル・ドライバの機能と処理内容の詳細，および実装している関数の仕様について説明します。

4.1 概　要

4.1.1 機　能

サンプル・ドライバには次のような処理が実装されています。

（1）初期化

USB ファンクション・コントローラを使用できるようにするため，各種レジスタを操作して設定します。大きく分けて，CPU レジスタに対する設定と USB ファンクション・コントローラのレジスタに対する設定があります。詳細は 4. 2. 1 CPU 初期化処理，　4. 2. 2 USB ファンクション・コントローラ初期化処理を参照してください。

（2）エンドポイントに対する処理

USB ファンクション・コントローラ内の転送用エンドポイントの状態は INTUSFA0I1 割り込みにより通知されます。大きく分けて，コントロール転送用エンドポイント（Endpoint0）に対してサンプル・ドライバでデコードを行うリクエストを受信した時の CPUDEC 割り込み，とバルク・アウト転送（受信）用エンドポイント（Endpoint2）に対してデータが正常受信されたことを示す BKO1DT 割り込みがあります。Endpoint0 の処理では，リクエスト応答も行います。詳細は 4. 2. 3 INTUSFA0I1 割り込み処理を参照してください。

（3）サンプル・アプリケーション

バルク・アウト転送（受信）用エンドポイントにあるデータを読み出し，読み出したデータをそのままバルク・イン転送（送信）用エンドポイントに書き込みます。詳細は第 5 章 サンプル・アプリケーションの仕様を参照してください。

4.1.2 リクエストへの対応

ここでは，サンプル・ドライバが対応する USB リクエストについて説明します。

(1) 標準リクエスト

V850E2/MN4 が自動的に応答しないリクエストに対し，サンプル・ドライバは次のような応答処理を行います。

(a) GET_DESCRIPTOR_string

ホストがファンクション・デバイスのストリング・ディスクリプタを取得するためのリクエストです。
このリクエストを受信すると，サンプル・ドライバは要求されたストリング・ディスクリプタの送信処理（コントロール・リード転送）を行います。

(b) SET_DESCRIPTOR

ホストがファンクション・デバイスのディスクリプタを設定するためのリクエストです。
このリクエストを受信すると，サンプル・ドライバは STALL 応答を返します。

(2) クラス・リクエスト

USB コミュニケーション・デバイス・クラス（CDC）のバルク・オンリー転送プロトコルのクラス・リクエストに対し，サンプル・ドライバ次のような応答処理を行います。

(a) SendEncapsulatedCommand

CDC インタフェースの制御プロトコルのフォーマットでコマンドを発行するためのリクエストです。
このリクエストを受信すると，サンプル・ドライバはリクエストに付随するデータを取り込み，送信処理（バルク・イン転送）を行います。

(b) GetEncapsulatedResponse

CDC インタフェースの制御プロトコルのフォーマットで応答を要求するためのリクエストです。
サンプル・ドライバは現在，このリクエストをサポートしていません。

(c) SetLineCoding

シリアル通信の通信フォーマットを指定するためのリクエストです。
このリクエストを受信すると，サンプル・ドライバはリクエストに付随するデータを取り込んで通信レートなどを設定し，NULL パケットの送信処理（コントロール・リード転送）を行います。

(d) GetLineCoding

デバイス側の現在の通信フォーマット設定を取得するためのリクエストです。
このリクエストを受信すると，サンプル・ドライバは通信レートなどの設定を読み出し，送信処理（コントロール・リード転送）を行います。

(e) SetControlLineState

RS-232/V.24 形式の制御信号のためのリクエストです。
このリクエストを受信すると，サンプル・ドライバは NULL パケットの送信処理（コントロール・リード転送）を行います。

4.1.3　ディスクリプタの設定

サンプル・ドライバでの各ディスクリプタの設定を次に示します。各ディスクリプタの設定は，ヘッダ・ファイル "usbf850_desc.h" に記述されています。

(1) デバイス・ディスクリプタ

GET_DESCRIPTOR_device リクエストに応答して送信されるディスクリプタです。
GET_DESCRIPTOR_device リクエストにはハードウエアが自動的に応答するため，設定内容は USB ファンクション・コントローラの初期化時に USFA0DDn レジスタ（n = 0-17）に格納されます。

表 4-1　デバイス・ディスクリプタの設定

フィールド	サイズ（バイト）	設定値	説　明
bLength	1	0x12	ディスクリプタのサイズ：18 バイト
bDescriptorType	1	0x01	ディスクリプタの種類：デバイス
bcdUSB	2	0x0200	USB 仕様リリース番号：USB 2.0
bDeviceClass	1	0x02	クラス・コード：CDC
bDeviceSubClass	1	0x00	サブクラス・コード：なし
bDeviceProtocol	1	0x00	プロトコル・コード：固有プロトコル未使用
bMaxPacketSize0	1	0x40	Endpoint0 の最大パケット・サイズ：64
idVendor	2	0x045B	ベンダ ID：Renesas Electronics
idProduct	2	0x0200	プロダクト ID：V850E2/MN4
bcdDevice	2	0x0001	デバイスのリリース番号：第 1 版
iManufacturer	1	0x01	製造者を表すストリング・ディスクリプタへのインデックス：1
iProduct	1	0x02	製品を表すストリング・ディスクリプタへのインデックス：2
iSerialNumber	1	0x03	デバイスの製造番号を表すストリング・ディスクリプタへのインデックス：3
bNumConfigurations	1	0x01	コンフィギュレーションの数：1

(2) コンフィギュレーション・ディスクリプタ

GET_DESCRIPTOR_configuration リクエストに応答して送信されるディスクリプタです。
GET_DESCRIPTOR_configuration リクエストにはハードウエアが自動的に応答するため，設定内容は USB ファンクション・コントローラの初期化時に USFA0CIEn レジスタ（n = 0-255）に格納されます。

表 4-2　コンフィギュレーション・ディスクリプタの設定

フィールド	サイズ（バイト）	設定値	説　　明
bLength	1	0x09	ディスクリプタのサイズ：9 バイト
bDescriptorType	1	0x02	ディスクリプタの種類：コンフィギュレーション
wTotalLength	2	0x0030	コンフィギュレーション，インタフェース，およびエンドポイント・ディスクリプタの総バイト数：48 バイト
bNumInterfaces	1	0x02	このコンフィギュレーションが持つインタフェースの数：2
bConfigurationValue	1	0x01	このコンフィギュレーションの識別番号：1
iConfiguration	1	0x00	このコンフィギュレーションを記述するストリング・ディスクリプタへのインデックス：0
bmAttributes	1	0x80	このコンフィギュレーションの特徴：バス・パワード，リモート・ウエイクアップなし
bMaxPower	1	0x1B	このコンフィギュレーションの最大消費電流：54 mA

(3) インタフェース・ディスクリプタ

GET_DESCRIPTOR_configuration リクエストに応答して送信されるディスクリプタです。
GET_DESCRIPTOR_configuration リクエストにはハードウエアが自動的に応答するため，設定内容は USB ファンクション・コントローラの初期化時に USFA0CIEn レジスタ（n = 0-255）に格納されます。
サンプル・ドライバではインタフェースを 2 つ使用するため，ディスクリプタも 2 種類設定しています。

表 4-3　Interface0 のインタフェース・ディスクリプタの設定

フィールド	サイズ（バイト）	設定値	説　　明
bLength	1	0x09	ディスクリプタのサイズ：9 バイト
bDescriptorType	1	0x04	ディスクリプタの種類：インタフェース
bInterfaceNumber	1	0x00	このインタフェースの識別番号：0
bAlternateSetting	1	0x00	このインタフェースに対するオルタナティブ設定の有無：なし
bNumEndpoints	1	0x01	このインタフェースが持つエンドポイントの数：1
bInterfaceClass	1	0x02	クラス・コード：コミュニケーション・インターフェース・クラス
bInterfaceSubClass	1	0x02	サブクラス・コード：Abstract Control Model
bInterfaceProtocol	1	0x00	プロトコル・コード：固有プロトコル使用せず
iInterface	1	0x00	このインタフェースを記述するストリング・ディスクリプタへのインデックス：0

表 4-4　Interface1 のインタフェース・ディスクリプタの設定

フィールド	サイズ（バイト）	設定値	説　明
bLength	1	0x09	ディスクリプタのサイズ：9 バイト
bDescriptorType	1	0x04	ディスクリプタの種類：インタフェース
bInterfaceNumber	1	0x01	このインタフェースの識別番号：1
bAlternateSetting	1	0x00	このインタフェースに対するオルタナティブ設定の有無：なし
bNumEndpoints	1	0x02	このインタフェースが持つエンドポイントの数：2
bInterfaceClass	1	0x0A	クラス・コード：コミュニケーション・インターフェース・クラス
bInterfaceSubClass	1	0x00	サブクラス・コード：Abstract Control Model
bInterfaceProtocol	1	0x00	プロトコル・コード：固有プロトコル使用せず
iInterface	1	0x00	このインタフェースを記述するストリング・ディスクリプタへのインデックス：0

(4) エンドポイント・ディスクリプタ

GET_DESCRIPTOR_configuration リクエストに応答して送信されるディスクリプタです。
GET_DESCRIPTOR_configuration リクエストにはハードウエアが自動的に応答するため，設定内容は USB ファンクション・コントローラの初期化時に USFA0CIEn レジスタ（n = 0-255）に格納されます。
サンプル・ドライバではエンドポイントを 3 つ使用するため，ディスクリプタも 3 種類設定しています。

表 4-5　Endpoint1（バルク・イン）のエンドポイント・ディスクリプタの設定

フィールド	サイズ（バイト）	設定値	説　明
bLength	1	0x07	ディスクリプタのサイズ：7 バイト
bDescriptorType	1	0x05	ディスクリプタの種類：エンドポイント
bEndpointAddress	1	0x81	このエンドポイントの転送方向：IN 方向 このエンドポイントのアドレス：1
bmAttributes	1	0x02	このエンドポイントの転送タイプ：バルク
wMaxPacketSize	2	0x0040	この転送の最大パケット・サイズ：64 バイト
bInterval	1	0x00	このエンドポイントのポーリング間隔：0 ms

表 4-6　Endpoint2（バルク・アウト）のエンドポイント・ディスクリプタの設定

フィールド	サイズ（バイト）	設定値	説　明
bLength	1	0x07	ディスクリプタのサイズ：7 バイト
bDescriptorType	1	0x05	ディスクリプタの種類：エンドポイント
bEndpointAddress	1	0x02	このエンドポイントの転送方向：OUT 方向 このエンドポイントのアドレス：2
bmAttributes	1	0x02	このエンドポイントの転送タイプ：バルク
wMaxPacketSize	2	0x0040	この転送の最大パケット・サイズ：64 バイト
bInterval	1	0x00	このエンドポイントのポーリング間隔：0 ms

表 4-7　Endpoint7（インタラプト・イン）のエンドポイント・ディスクリプタの設定

フィールド	サイズ（バイト）	設定値	説　　明
bLength	1	0x07	ディスクリプタのサイズ：7 バイト
bDescriptorType	1	0x05	ディスクリプタの種類：エンドポイント
bEndpointAddress	1	0x87	このエンドポイントの転送方向：IN 方向 このエンドポイントのアドレス：7
bmAttributes	1	0x03	このエンドポイントの転送タイプ：インタラプト
wMaxPacketSize	2	0x0008	この転送の最大パケット・サイズ：8 バイト
bInterval	1	0x0A	このエンドポイントのポーリング間隔：10 ms

(5) ストリング・ディスクリプタ

GET_DESCRIPTOR_string リクエストに応答して送信されるディスクリプタです。
GET_DESCRIPTOR_string リクエストを受信すると，サンプル・ドライバはこのディスクリプタの設定をヘッダ・ファイル "usbf850_desc.h" から取り出して，USB ファンクション・コントローラの USFA0E0W レジスタに格納します。

表 4-8　ストリング・ディスクリプタの設定

（a）String 0

フィールド	サイズ（バイト）	設定値	説　　明
bLength	1	0x04	ディスクリプタのサイズ：4 バイト
bDescriptorType	1	0x03	ディスクリプタの種類：ストリング
bString	2	0x09, 0x04	言語コード：英語（U.S.）

（b）String 1

フィールド	サイズ（バイト）	設定値	説　　明
bLength 注6	1	0x30	ディスクリプタのサイズ：42 バイト
bDescriptorType	1	0x03	ディスクリプタの種類：ストリング
bString 注7	46	–	ベンダ：Renesas Electronics Co.

（注 6）　bString フィールドのサイズにより設定値が異なります。
（注 7）　ベンダにより任意に設定できる領域のため，サイズや設定値は一定ではありません。

（c）String 2

フィールド	サイズ（バイト）	設定値	説　　明
bLength 注8	1	0x0E	ディスクリプタのサイズ：14 バイト
bDescriptorType	1	0x03	ディスクリプタの種類：ストリング
bString 注9	12	–	製品の種類：CDCDrv（CDC ドライバ）

（注 8）　bString フィールドのサイズにより設定値が異なります。
（注 9）　ベンダにより任意に設定できる領域のため，サイズや設定値は一定ではありません。

（d）String 3

フィールド	サイズ（バイト）	設定値	説　　明
bLength 注10	1	0x16	ディスクリプタのサイズ：24 バイト
bDescriptorType	1	0x03	ディスクリプタの種類：ストリング
bString 注11	22	–	シリアル番号： V850E2/MN4 ： 020002020010

（注 10）　bString フィールドのサイズにより設定値が異なります。

（注 11）　ベンダにより任意に設定できる領域のため，サイズや設定値は一定ではありません。

4.2　各部の動作

サンプル・ドライバを実行すると，次のような一連の処理を行います。ここでは，それぞれの処理について説明します。

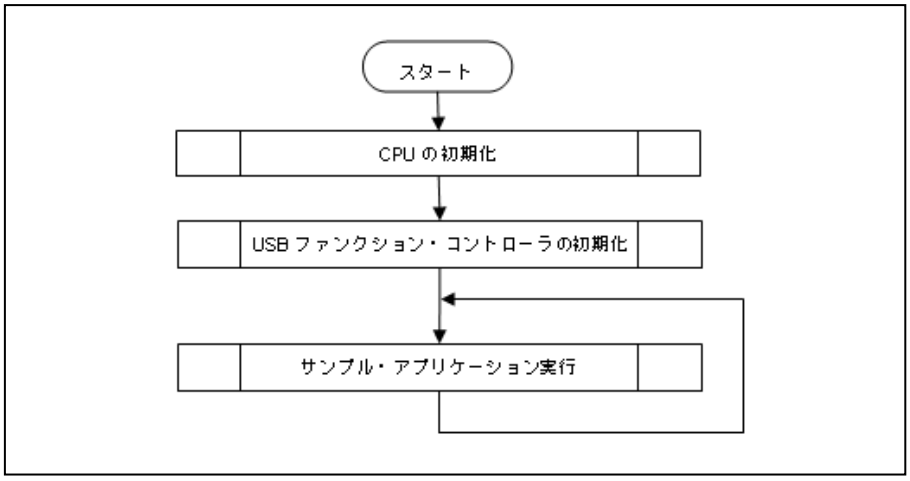


図 4−1　サンプル・ドライバの処理フロー

4.2.1　CPU初期化処理

USB ファンクション・コントローラを使用するために必要な項目を設定します。

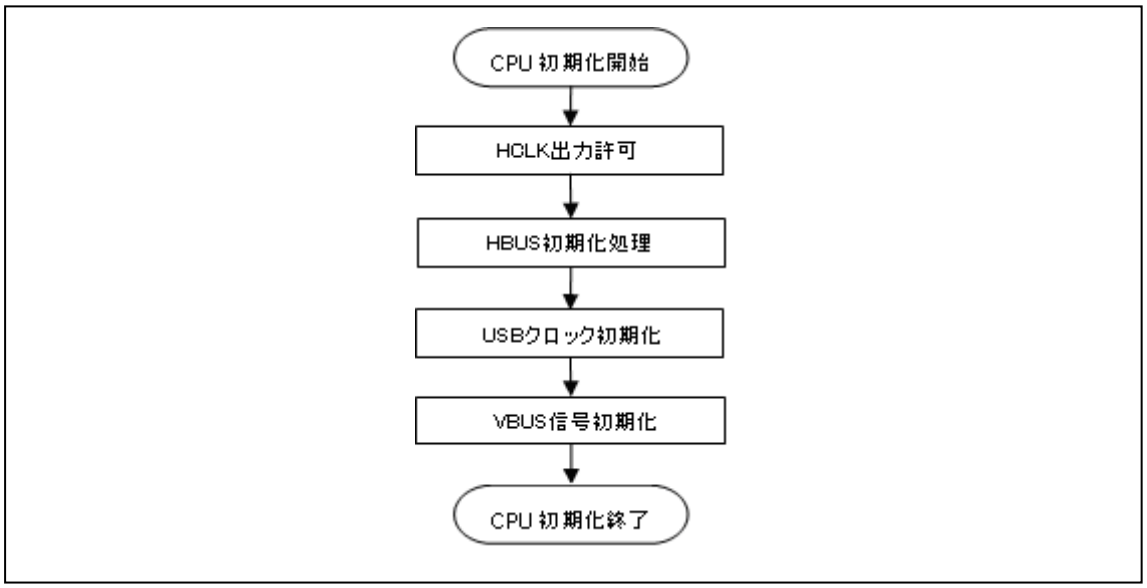


図 4－2　CPU 初期化の処理フロー

(1) HCLK 出力許可設定

HCLK の出力を許可し, H バスに接続される USBF が動作できるように設定します。設定を行う SFRCTL2 レジスタは特定書き込みレジスタとなっているため, 特定書き込みシーケンスを行って設定を行います。

(2) H バス初期化処理

H バスの初期化処理を行います。指定の指示に従い, H バスの初期化を行います。詳細は V850E2/MN4マイクロコントローラのユーザーズマニュアル　ハードウエア編を参照してください。

(3) USB クロックの初期化

UCLK が接続されている兼用端子 P13 の設定を行います。本サンプルでは USB クロックとして UCLK を使用しており，これにより USB クロック入力が行われます。

(4) VBUS 信号初期化

VBUS 信号の初期化設定を行います。

4.2.2　USBファンクション・コントローラ初期化処理

USB ファンクション・コントローラの使用を開始するために必要な項目を設定します。

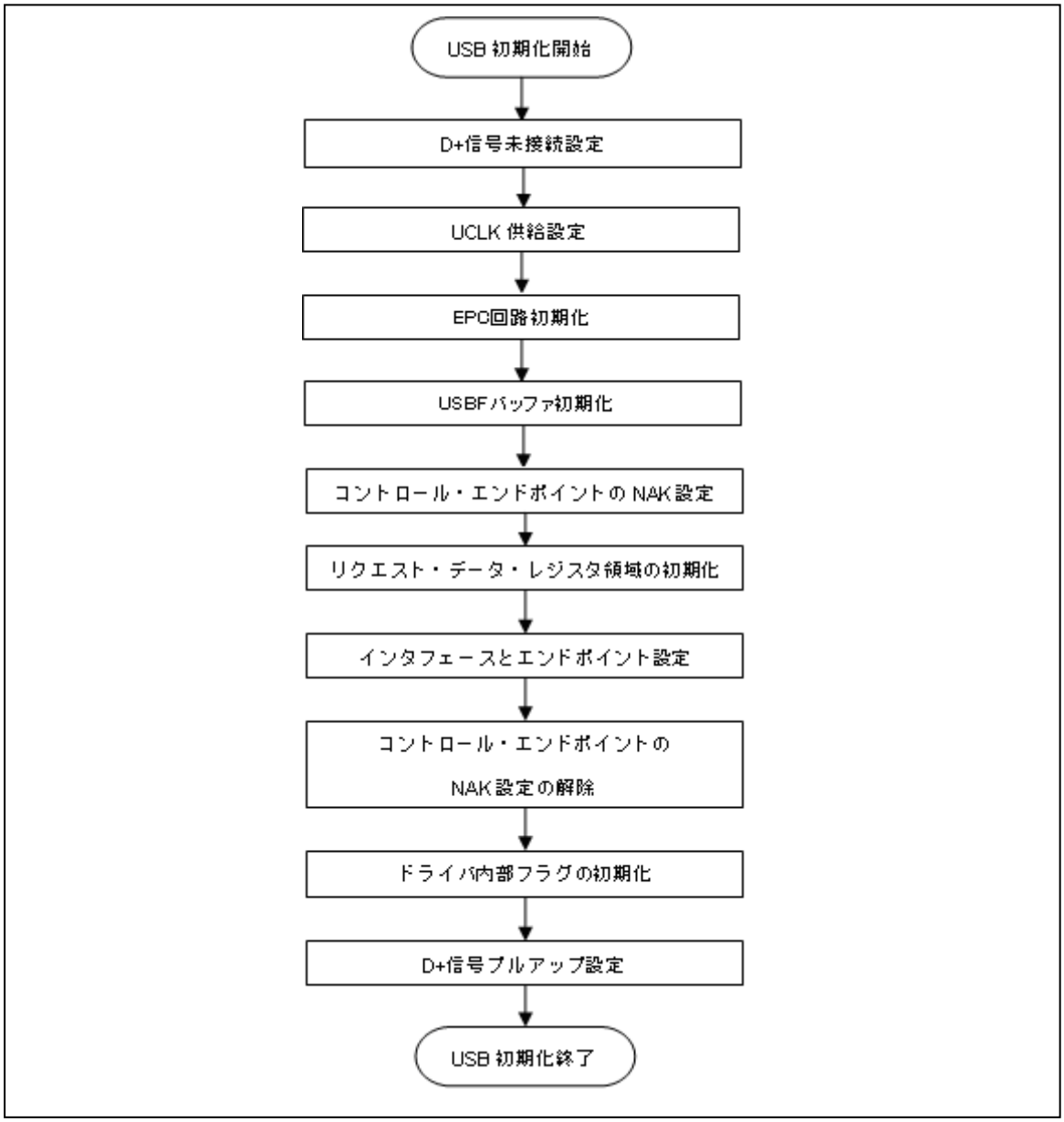


図 4−3　USB ファンクション・コントローラ初期化の処理フロー

(1) D+信号プルダウン設定

CPU の P4.10 に”0”を設定します。これにより，D+信号をロウ･レベルの出力にして，ホスト側にデバイスの接続を検知されない様にします。

(2) UCLK 供給設定

SFRCTL3 レジスタに”0x48”を設定し，USB ファンクションへのクロック供給を許可します。

(3) EPC 回路初期化

USFA0EPCCTL レジスタに”0x00000000”を設定し，EPC リセット信号を解除します。

(4) USB ファンクション・バッファ初期化

USFBC レジスタに”0x00000003”を設定し，USBF バッファ有効及びフローティング対策有効設定を行います。

(5) コントロール・エンドポイントの NAK 設定

ここでは USFA0E0NA レジスタの EP0NKA ビットに "1" を書き込みます。この設定により，自動応答リクエストを含むすべてのリクエストに対してハードウエアが NAK で応答します。

このビットは，自動応答リクエストで使用するデータの登録が完了するまで，ハードウエアが自動応答リクエストに対して意図しないデータを返さないようにするために，ソフトウエアが使用します。

(6) リクエスト・データ・レジスタ領域の初期化

GET_DESCRIPTOR リクエストに自動応答するためのディスクリプタ・データなどを各種レジスタに登録します。

ここでは次に示すレジスタにアクセスします。

(a) USFA0DSTL レジスタに "0x01" を書き込みます。この設定により，リモート・ウエイクアップ機能の使用が禁止され，USB ファンクション・コントローラはセルフ・パワード・デバイスとして動作します。

(b) USFA0EnSL レジスタ（n = 0-2）に "0x00" を書き込みます。この設定により，Endpoint n が正常に動作していることを示します。

(c) USFA0DSCL レジスタに，必要なディスクリプタのデータ長の合計（バイト数）を書き込みます。この設定により，使用される USFA0CIEn レジスタ（n = 0-255）の範囲が決まります。

(d) USFA0DDn レジスタ（n = 0-7）にデバイス・ディスクリプタのデータを書き込みます。

(e) USFA0CIEn レジスタ（n = 0-255）にコンフィギュレーション・ディスクリプタ，インタフェース・ディスクリプタ，およびエンドポイント・ディスクリプタのデータを書き込みます。

(f) USFA0MODC レジスタに "0x00" を書き込みます。この設定により，GET_DESCRIPTOR_configuration リクエストへの自動応答が許可されます。

(7) インタフェースとエンドポイントの設定

サポートするインタフェースの数，オルタナティブ設定の状態，インタフェースとエンドポイントの関係などの情報を各種レジスタに設定します。

ここでは次に示すレジスタにアクセスします。

(a) USFA0AIFN レジスタに "0x80" を書き込みます。この設定により，2 つまでのインタフェースを有効にします。

(b) USFA0AAS レジスタに "0x00" を書き込みます。この設定により，オルタナティブ設定を無効にします。

(c) USFA0E1IM レジスタに "0x40" を書き込みます。この設定により，Endpoint1 が Interface1 にリンクされます。

(d) USFA0E2IM レジスタに "0x40" を書き込みます。この設定により，Endopoint2 が Intreface1 にリンクされます。

(e) USFA0E7IM レジスタに "0x20" を書き込みます。この設定により，Endopoint7 が Intreface0 にリンクされます。

(8) コントロール・エンドポイントの NAK 設定の解除

ここでは USFA0E0NA レジスタの EP0NKA ビットに "0" を書き込みます。この設定により，自動応答リクエストを含むすべてのリクエストに対して，それぞれに応じた応答が再開されます。

(9) 割り込みマスク・レジスタの設定

USB ファンクション・コントローラの割り込み要因ごとのマスクを設定します。
ここでは次に示すレジスタにアクセスします。

(a) USFA0ICn レジスタ（n = 0-4）に "0x00" を書き込みます。この設定により，すべての割り込み要因がクリアされます。
(b) USFA0FIC0 レジスタに "0xF7" を，USFA0FIC1 レジスタに”0x0F”を書き込みます。この設定により，すべての転送用 FIFO がクリアされます。
(c) USFA0IM0 レジスタに "0x1B" を書き込みます。この設定により，USFA0IS0 レジスタに示される割り込み要因のうち，BUSRST 割り込み，RSUSPD 割り込み，SETRQ 割り込み以外の要因がすべてマスクされます。
(d) USFA0IM1 レジスタに "0x7E" を書き込みます。この設定により，USFA0IS1 レジスタに示される割り込み要因のうち，CPUDEC 割り込み以外の要因がすべてマスクされます。
(e) USFA0IM2 レジスタに "0xF1" を書き込みます。この設定により，USFA0IS2 レジスタに示される割り込み要因がすべてマスクされます。
(f) USFA0IM3 レジスタに "0xFE" を書き込みます。この設定により，USFA0IS3 レジスタに示される割り込み要因のうち，BKO1DT 割り込み以外の要因がすべてマスクされます。
(g) USFA0IM4 レジスタに "0x20" を書き込みます。この設定により，USFA0IS4 レジスタに示される割り込み要因がすべてマスクされます。
(i) USFA0EPCINTE レジスタに”0x0003”を書き込み，EPC_INT0BEN，EPC_INT1BEN ビットが立った時の割り込みを有効にします。
(j) ICUSFA0I1 に”0”を，ICUSFA0I2 に”0”を書き込み，INTUSFA0I1・INTUSFA0I2 を有効にします。

(10) ドライバ内部フラグの初期化

ドライバ内部で使用するフラグ(usbf850_busrst_flg，usbf850_rsuspd_flg，usbf850_rdata_flg)の初期化を行います。

(11) D+信号プルアップ設定

CPU の P4 レジスタに "0x0400" を書き込みます。この設定により，P4_10 から "1" が出力されます。これにより，D+信号からハイ・レベルを出力して，ホスト側にデバイスが接続されたことを通知します。サンプル・ドライバでは図 4−4 に示すような接続を想定しています。

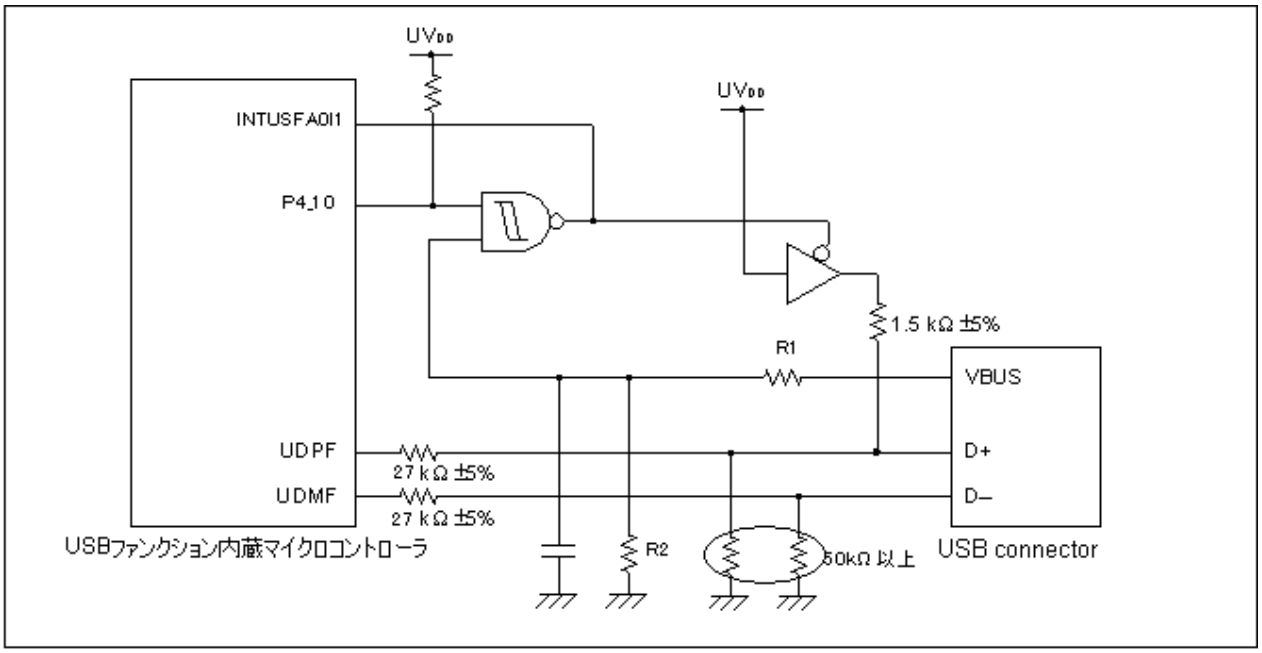


図 4−4　USB ファンクション・コントローラ接続例

4.2.3 USBF割り込み処理（INTUSFA0I1）

USB ファンクション・コントローラからの割り込み要求（INTUSFA0I1）は、初期化時にマスク解除された割り込みについてのみ通知されます。必要な割り込みについては、初期化時に割り込みマスクを解除して下さい。通知された割り込みについて、それぞれ必要な処理を行います。

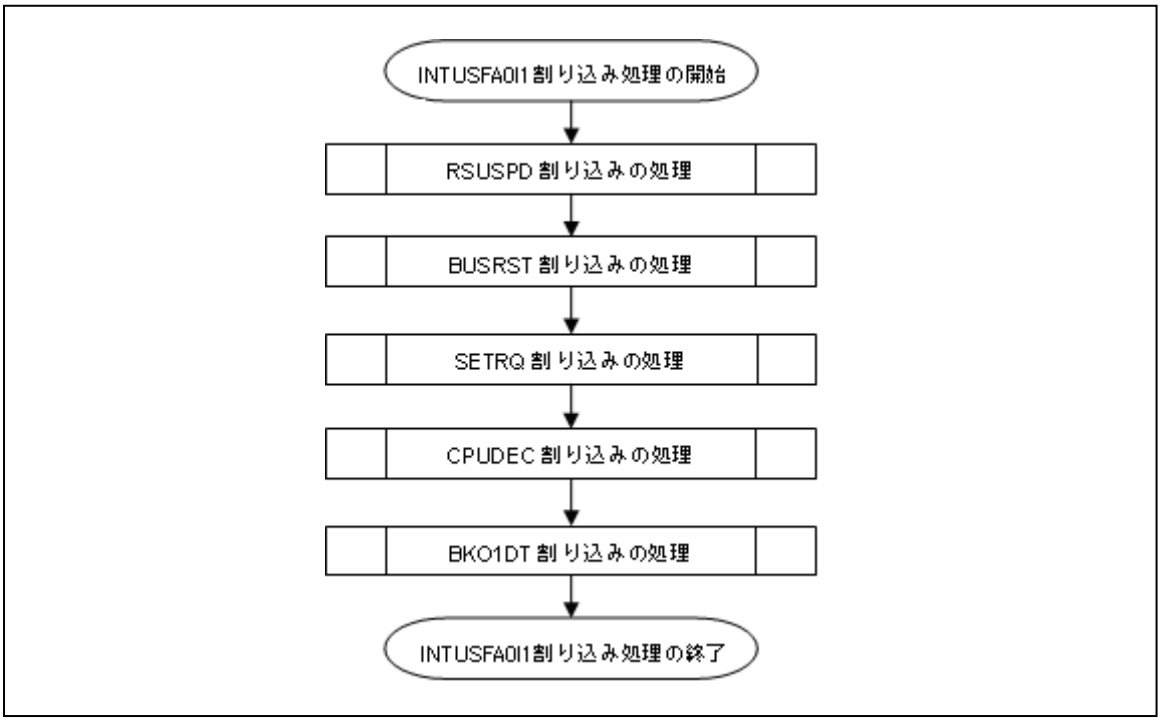


図 4－5　INTUSFA0I1 割り込みハンドラ処理フロー

（1）RSUSPD 割り込みの処理

USFA0IS0 レジスタの RSUSPD ビットが "1" であれば、USFA0IC0 の RSUSPDC ビットをクリア(0)します。次に、USFA0EPS1 の RSUM ビットが”1”であれば、サスペンド状態を示すフラグ(usbf850_rsuspd_flg)を更新(サスペンド発生)します。

（2）BUSRST 割り込みの処理

USFA0IS0 レジスタの BUSRST ビットが "1"であれば、USFA0IC0 の BUSRSTC ビットをクリア(0)します。次に、バスリセットが発生した事を示すフラグ(usbf850_busrst_flg)を更新(バスリセット発生)し、usbf850_buff_init()関数を呼び出し、バッファを初期化します。

（3）SETRQ 割り込みの処理

USFA0IS0 レジスタの SETRQ ビットが "1"であれば、USFA0IC0 の SETRQC ビットをクリア(0)します。次に、USFA0SET レジスタの SETCON ビットと USFA0MODS レジスタの CONF ビットが”1”だった場合、Set Configuration リクエストが処理された事を示すフラグ(usbf850_busrst_flg)をクリアします。

（4）CPUDEC 割り込みの処理

USFA0IS1 レジスタの CPUDEC ビットが "1"であれば、USFA0IC1 の CPUDECC ビットをクリア(0)します。次に、USFA0E0ST レジスタを 8 回読み出し、リクエスト・データを取り込んでデコードします。リクエストが標準リクエストなら usbf850_standardreq()関数を呼び出し、クラス・リクエストなら usbf850_classreq()関数を呼び出します。

（5）BKO1DT 割り込みの処理

USFA0IS3 レジスタの BKODT ビットが "1"であれば、USFA0IC3 の BKODTC ビットをクリア(0)します。次に、データを受信した事を示すフラグ(usbf850_rdata_flg)を更新します。受信 FIFO に有効なデータが格納された場合に発生します。

4.3 関数の仕様

ここでは，サンプル・ドライバに実装されている各種関数について説明します。

4.3.1 関数一覧

サンプル・ドライバでは，ソース・ファイルそれぞれに次のような関数が実装されています。

表 4-9　サンプル・ドライバ内の関数

ソース・ファイル	関数名	説　　明
main.c	main	メイン・ルーチン
	cpu_init	CPUの初期化
	SetProtectReg	書き込み保護レジスタアクセス処理
usbf850.c	usbf850_init	USBファンクション・コントローラの初期化
	usbf850_intusbf0	Endpoint0の監視とリクエストへの応答制御
	usbf850_intusbf1	レジューム割り込み処理
	usbf850_data_send	USBデータの送信
	usbf850_data_receive	USBデータの受信
	usbf850_rdata_length	USB受信データ長の取得
	usbf850_send_EP0	Endpoint0の送信
	usbf850_receive_EP0	Endpoint0の受信
	usbf850_send_null	Bulk/ Interrupt In EndpointへのNullパケット送信処理
	usbf850_sendnullEP0	Endpoint0用NULLパケットの送信
	usbf850_sendstallEP0	Endpoint0用STALL応答
	usbf850_ep_status	Bulk/ Interrupt In EndpointのFIFO状態通知処理
	usbf850_fifo_clear	Endpoint0以外のEndpointのFIFOクリア
	usbf850_standardreq	標準リクエストの処理
	usbf850_getdesc	GET_DESCRIPTORリクエストの処理
usbf850_communication.c	usbf850_classreq	CDCクラス・リクエストの処理
	usbf850_send_encapsulated_command	Send Encapsulated Commandリクエストの処理
	usbf850_get_encapsulated_response	Get Encapsulated Commandリクエストの処理
	usbf850_set_line_coding	Set Line Codingリクエストの処理
	usbf850_get_line_coding	Get Line Codingリクエストの処理
	usbf850_set_control_line_state	Set Control Line Stateリクエストの処理
	usbf850_buff_init	CDCデータ転送用のEndpointのFIFOクリア処理
	usbf850_get_bufinit_flg	FIFO初期化処理の実行状態通知処理
	usbf850_send_buf	CDCデータの送信処理
	usbf850_recv_buf	CDCのクラス・リクエスト処理関数登録

4.3.2　関数の相関関係

関数によっては，処理の中で別の関数を呼び出しているものもあります。関数の呼び出し関係を次に示します。

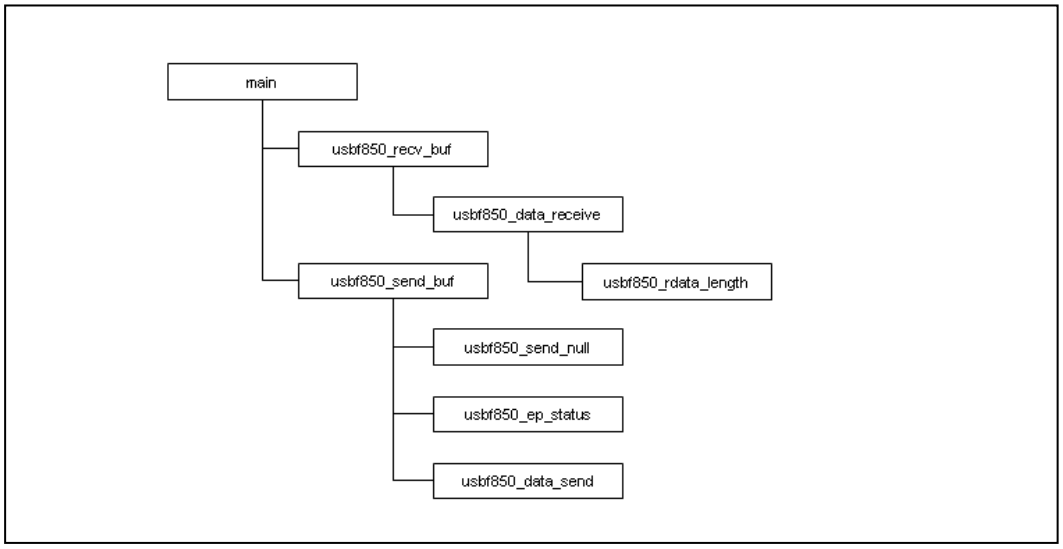


図 4−6　メイン・ルーチンでの関数の呼び出し

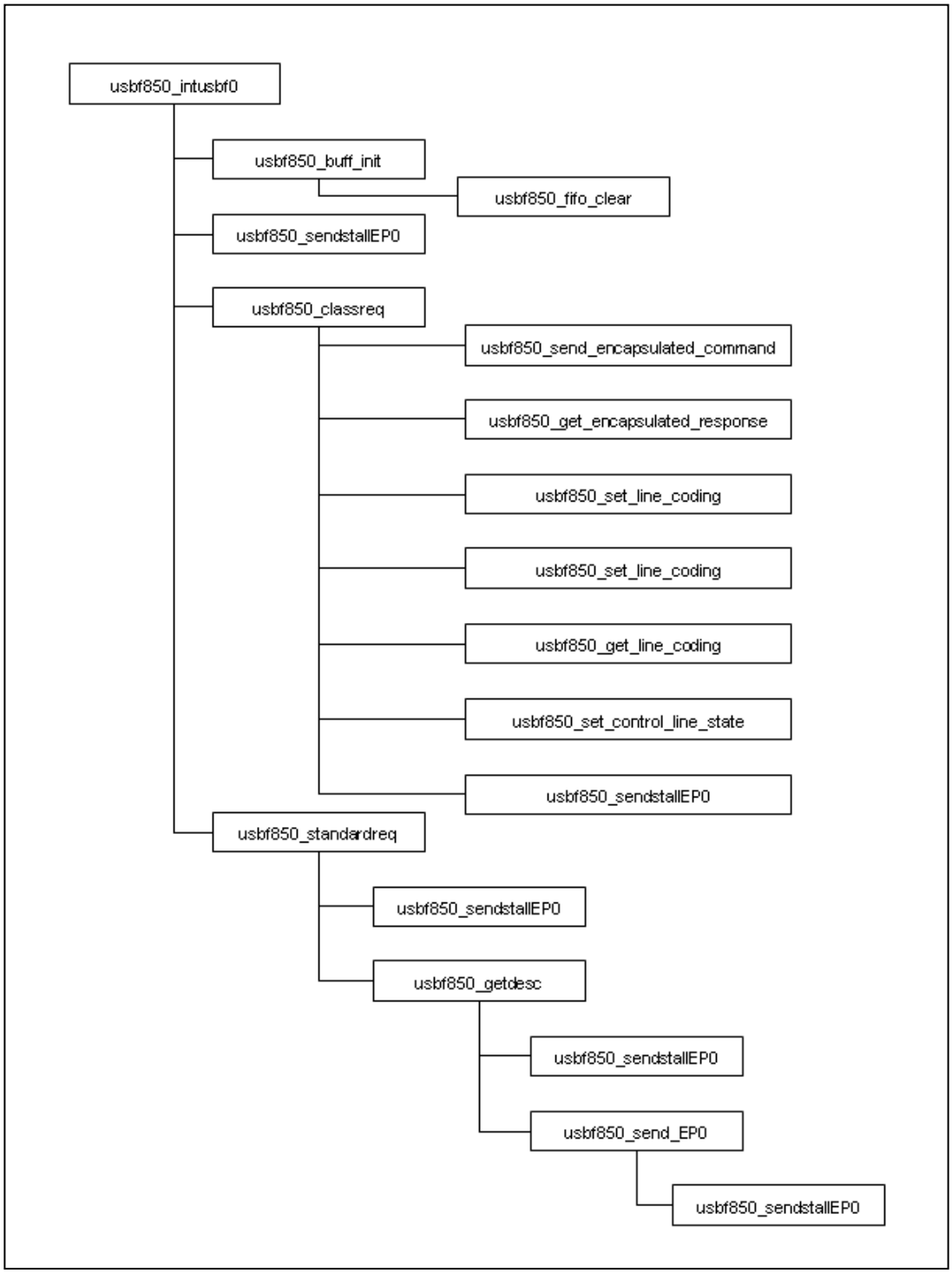


図 4−7　USB ファンクション・コントローラ用処理での関数の呼び出し

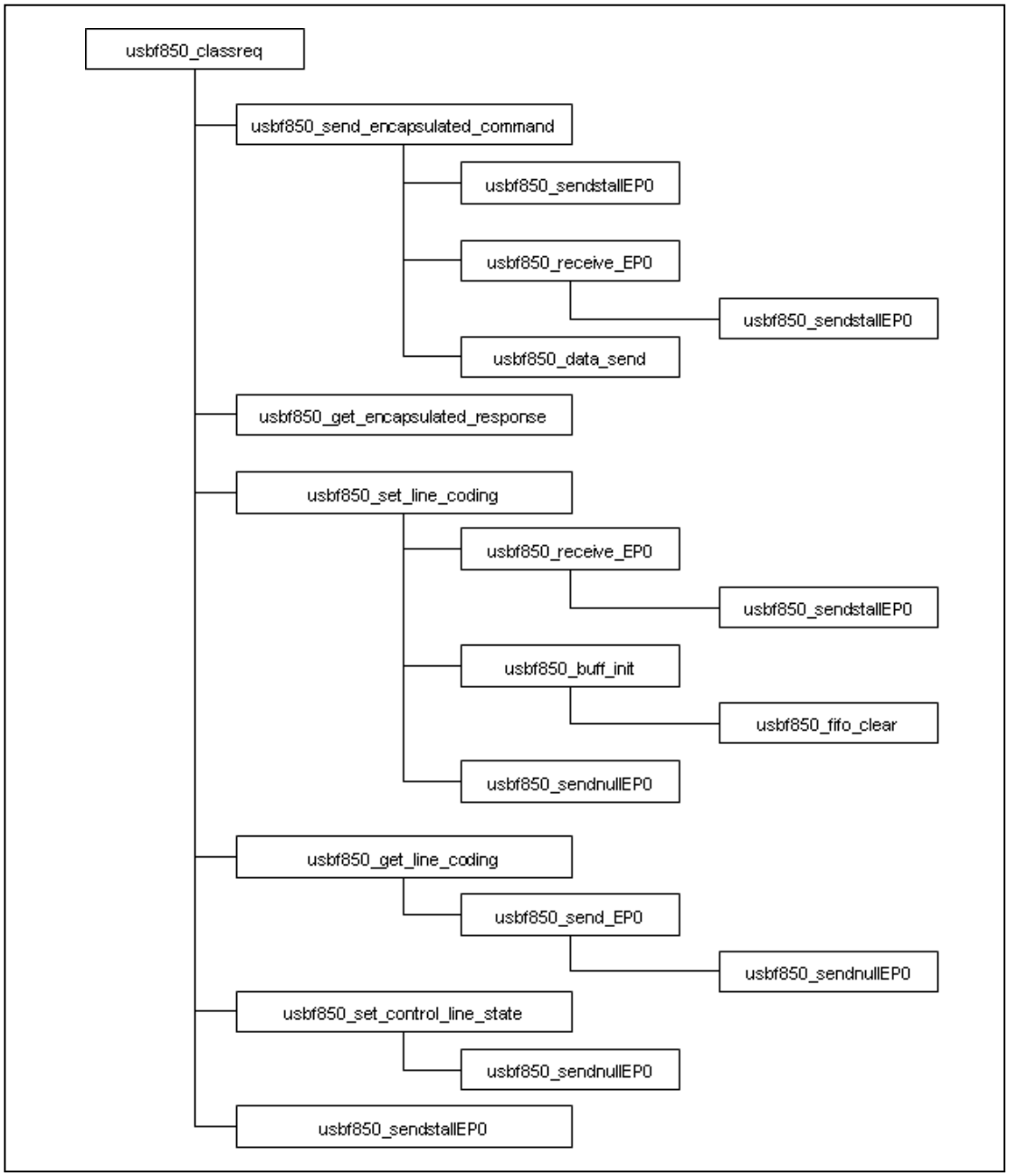


図 4−8　USB コミュニケーション・クラス用処理での関数の呼び出し(1/2)

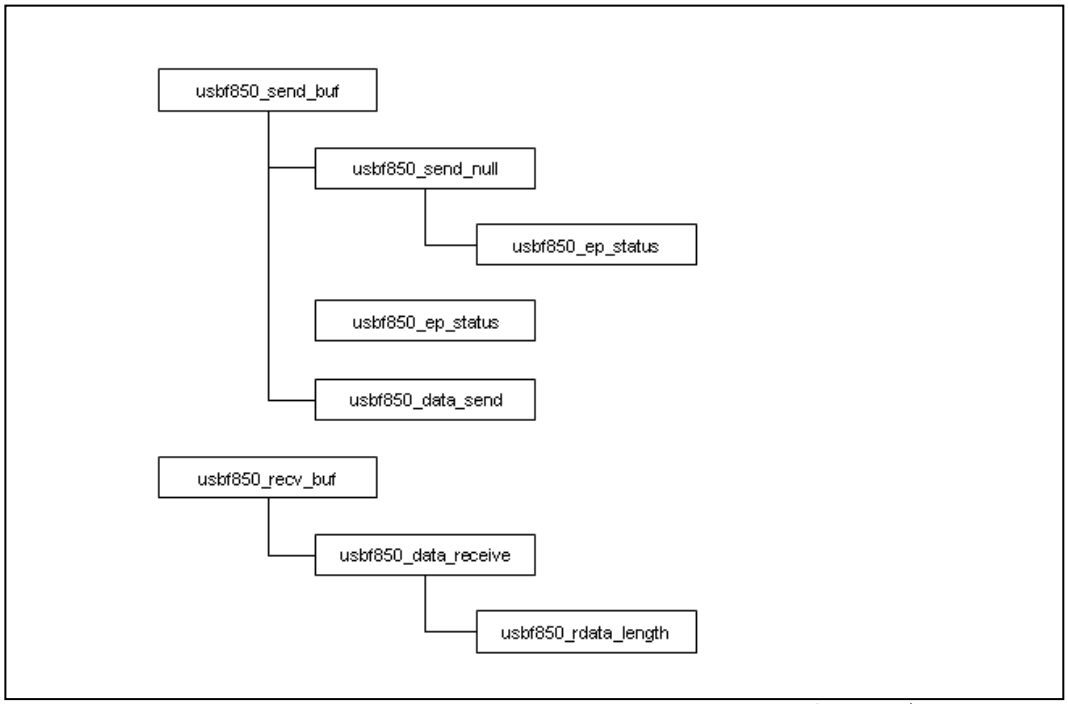


図 4－9　USB コミュニケーション・クラス用処理での関数の呼び出し(2/2)

4.3.3　関数の機能

ここでは，サンプル・ドライバに実装されている各種関数について解説します。

（1）関数解説フォーマット

解説は，関数ごとに次の形式で記述されます。

関数名称

【 概　要 】

概要説明

【C 言語記述形式】

C 言語上の記述形式

【パラメータ】

パラメータ（引数）の説明

パラメータ	説　　明
パラメータ型，名称	*パラメータ概要説明*

【 戻り値 】

戻り値の説明

シンボル	説　　明
戻り値型，名称	*戻り値概要説明*

【 機　能 】

機能説明

(2) メイン・ルーチンの関数

main

【 概　要 】
メイン処理

【C 言語記述形式】
void main(void)

【パラメータ】
なし

【 戻り値 】
なし

【 機　能 】
サンプル・ドライバを実行すると最初に呼び出される関数です。
CPU、USB ファンクション・コントローラの初期化を順に実行します。次に，サンプル・アプリケーションの処理とサスペンドが発生を示すフラグ(usbf850_rsuspd_flg)の監視を行います。サスペンドが発生した場合、ここで、サスペンドとレジュームの処理を行います。

cpu_init

【 概　要 】
CPU 初期化処理

【C 言語記述形式】
void cpu_init(void)

【パラメータ】
なし

【 戻り値 】
なし

【 機　能 】
初期化処理で呼び出される関数です。
H バス初期化や USB クロックなど，USB ファンクション・コントローラを使用するために必要な項目等を設定します。

SetProtectReg

【 概　要 】

書き込み保護レジスタへのアクセス

【C 言語記述形式】

void SetProtectReg(volatile UINT32 *dest_reg, UINT32 wr_dt, volatile UINT8 *prot_reg)

【パラメータ】

パラメータ	説　　明
volatile UINT32 *dest_reg	保護されたレジスタアドレス
UINT32 wr_dt	書き込み数値
volatile UINT8 *prot_reg	保護コマンドレジスタアドレス

【 戻り値 】

なし

【 機　能 】

書き込み保護されたレジスタへの書き込み処理を行います。

（3）USB ファンクション・コントローラ用処理の関数

usbf850_init

【 概　要 】
USB ファンクション・コントローラ初期化処理

【C 言語記述形式】
void usbf850_init(void)

【パラメータ】
なし

【 戻り値 】
なし

【 機　能 】
初期化処理で呼び出される関数です。
データ領域の確保と設定，割り込み要求のマスクなど，USB ファンクション・コントローラの使用を開始するために必要な項目を設定します。

usbf850_intusbf0

【 概　要 】

Endpoint0 監視処理
Endpoint2 監視処理

【C 言語記述形式】

void usbf850_intusbf0(void)

【パラメータ】

なし

【 戻り値 】

なし

【 機　能 】

INTUSFA0I1 割り込みにより呼び出される割り込みサービスルーチンです。
USB ファンクション・コントローラのマスクされていない割り込みについて、割り込み内容を確認し、発生している割り込みについて処理を行います。

usbf850_intusbf1

【 概　要 】
Resume 割り込み処理

【C 言語記述形式】
void usbf850_intusbf1(void)

【パラメータ】
なし

【 戻り値 】
なし

【 機　能 】
INTUSFA0I2 割り込みにより呼び出される割り込みサービスルーチンです。サスペンド状態を示すフラグをクリアして、レジューム状態に更新します。

usbf850_data_send

【 概　要 】

Bulk/ Interrupt In Endpoint 用 USB データ送信処理

【C 言語記述形式】

INT32 usbf850_data_send(UINT8 *data, INT32 len, INT8 ep)

【パラメータ】

パラメータ	説　　明
UINT8 *data	送信データ・バッファ・ポインタ
INT32 len	送信データ長
INT8 ep	データ送信エンドポイント番号

【 戻り値 】

シンボル	説　　明
len（>=0）	正常に送信したデータサイズ
DEV_ERROR	異常終了

【 機　能 】

送信データ・バッファに格納されているデータを，指定されたエンドポイント用の FIFO に 1 バイトずつ格納します。

usbf850_data_receive

【 概　要 】

USB データ受信処理

【C 言語記述形式】

INT32 usbf850_data_receive(UINT8 *data, INT32 len, INT8 ep)

【パラメータ】

パラメータ	説　　明
UINT8 *data	受信データ・バッファ・ポインタ
INT32 len	受信データ長
INT8 ep	データ受信エンドポイント番号

【 戻り値 】

シンボル	説　　明
len（>=0）	正常に送信したデータ・サイズ
DEV_ERROR	異常終了

【 機　能 】

指定されたエンドポイント用の FIFO からデータを 1 バイトずつ読み出し，受信データ・バッファに格納します。

usbf850_rdata_length

【 概　要 】
USB データ受信データ長取得

【C 言語記述形式】
void usbf850_rdata_length(INT32 *len , INT8 ep)

【パラメータ】

パラメータ	説　　明
INT32* len	受信データ長格納アドレス・ポインタ
INT8 ep	データ受信エンドポイント番号

【 戻り値 】
なし

【 機　能 】
指定されたエンドポイントの受信データ長を読み出します。

usbf850_send_EP0

【 概　要 】

Endpoint0 用 USB データ送信処理

【C 言語記述形式】

INT32 usbf850_send_EP0(UINT8* data, INT32 len)

【パラメータ】

パラメータ	説　　明
UINT* data	送信データ・バッファ・ポインタ
INT32 len	送信データ・サイズ

【 戻り値 】

シンボル	説　　明
DEV_OK	正常終了
DEV_ERROR	異常終了

【 機　能 】

送信データ･バッファに格納されているデータを Endpoint0 用送信 FIFO に 1 バイトずつ格納します。

usbf850_receive_EP0

【 概　要 】

Endpoint0 用 USB データ受信処理

【C 言語記述形式】

INT32 usbf850_receive_EP0(UINT8* data, INT32 len)

【パラメータ】

パラメータ	説　　明
UINT* data	受信データ・バッファ・ポインタ
INT32 len	受信データ・サイズ

【 戻り値 】

シンボル	説　　明
DEV_OK	正常終了
DEV_ERROR	異常終了

【 機　能 】

Endpoint0 用受信 FIFO から 1 バイトずつ読み出し，受信データ・バッファに格納します。

usbf850_send_null

【 概　要 】

Bulk/ Interrupt In Endpoint 用 Null パケット送信処理

【C 言語記述形式】

INT32 usbf850_send_null(INT8 ep)

【パラメータ】

パラメータ	説　　明
INT8 ep	データ送信エンドポイント番号

【 戻り値 】

シンボル	説　　明
DEV_OK	正常終了
DEV_ERROR	異常終了

【 機　能 】

指定された Endpoint（送信用）の FIFO をクリアし，データ終了を示すビットをセット(1)する事で，USB ファンクション・コントローラから Null パケットを送信します。

usbf850_sendnullEP0

【 概　要 】

Endpoint0 用 NULL パケット送信処理

【C 言語記述形式】

void usbf850_sendnullEP0(void)

【パラメータ】

なし

【 戻り値 】

なし

【 機　能 】

Endpoint0 用の FIFO をクリアし，データ終了を示すビットをセット（1）することで，USB ファンクション・コントローラから NULL パケットを送信させます。

usbf850_sendstallEP0

【 概　要 】
Endpoint0 用 STALL 応答処理

【C 言語記述形式】
void usbf850_sendstallEP0(void)

【パラメータ】
なし

【 戻り値 】
なし

【 機　能 】
STALL ハンドシェーク使用を示すビットをセット（1）することで，USB ファンクション・コントローラから STALL 応答させます。

usbf850_ep_status

【 概　要 】

Bulk/ Interrupt In Endpoint 用 FIFO 状態通知処理

【C 言語記述形式】

INT32 usbf850_ep_status(INT8 ep)

【パラメータ】

パラメータ	説　　明
INT8 ep	データ送信Endpoint番号

【 戻り値 】

シンボル	説　　明
DEV_OK	正常終了
DEV_RESET	Bus Reset処理中
DEV_ERROR	異常終了

【 機　能 】

指定された Endpoint(送信用)の FIFO 状態を通知します。

usbf850_fifo_clear

【 概　要 】

Bulk/ Interrupt Endpoint 用 FIFO クリア処理

【C 言語記述形式】

void usbf850_fifo_clear(INT8 in_ep, INT8 out_ep)

【パラメータ】

パラメータ	説　　明
INT8 in_ep	データ送信Endpoint
INT8 out_ep	データ受信Endpoint

【 戻り値 】

なし

【 機　能 】

指定された Endpoint（Bulk/Interrupt）の FIFO をクリアし，データ受信フラグ(usbf850_rdata_flg)をクリアします。

usbf850_standardreq

【 概　要 】

USB ファンクション・コントローラが自動応答しない標準リクエストの処理

【C 言語記述形式】

void usbf850_standardreq(USB_SETUP *req_data)

【パラメータ】

パラメータ	説　　明
USB_SETUP *req_data	受信データ長格納アドレス・ポインタ

【 戻り値 】

なし

【 機　能 】

INTUSFA0I1 割り込み処理の CPUDEC 割り込み要因の場合に呼び出される関数です。
デコードされたリクエストが GET_DESCRIPTOR の場合，GET_DESCRIPTOR リクエスト処理関数（usbf850_getdesc）を呼び出します。それ以外のリクエストの場合は Endpoint0 用 STALL 応答処理関数（usbf850_sendstallEP0）を呼び出します。

usbf850_getdesc

【 概　要 】

GET_DESCRIPTOR リクエスト処理

【C 言語記述形式】

void usbf850_getdesc(USB_SETUP *req_data)

【パラメータ】

パラメータ	説　　明
USB_SETUP *req_data	受信データ長格納アドレス・ポインタ

【 戻り値 】

なし

【 機　能 】

USB ファンクション・コントローラが自動応答しない標準リクエストの処理で呼び出される関数です。

デコードされたリクエストがストリング・ディスクリプタを要求している場合，USB データ送信処理関数（usbf850_data_send）を呼び出して，Endpoint0 からストリング・ディスクリプタを送信させます。それ以外のディスクリプタを要求している場合は Endpoint0 用 STALL 応答処理関数（usbf850_sendstallEP0）を呼び出します。

（4）USB コミュニケーション・クラス用処理の関数

usbf850_classreq

【 概　要 】

CDC のクラス・リクエスト処理

【C 言語記述形式】

void usbf850_classreq(USB_SETUP *req_data)

【パラメータ】

パラメータ	説　　明
USB_SETUP *req_data	リクエスト・データ格納ポインタ・アドレス

【 戻り値 】

なし

【 機　能 】

INTUSFA0I1 割り込み処理の CPUDEC 割り込み要因で呼び出される関数です。デコードされたリクエストがコミュニケーション･デバイス・クラス固有のリクエストの場合、各リクエスト処理関数を呼び出します。それ以外の場合は Endpoint0 に Stall を送信します。

usbf850_send_encapsulated_command

【 概　要 】

Send Encapsulated Command リクエスト処理

【C 言語記述形式】

void usbf850_send_encapsulated_command(void)

【パラメータ】

なし

【 戻り値 】

なし

【 機　能 】

データ受信処理関数（usbf850_data_receive）を呼び出して Endpoint0 で受信したデータを取り込み，データ送信処理関数（usbf850_data_send）を呼び出して Endpoint2 からバルク・イン転送（送信）でデータを送信させます。

usbf850_set_line_coding

【 概　要 】
Set Line Coding リクエスト処理

【C 言語記述形式】
void usbf850_set_line_coding(void)

【パラメータ】
なし

【 戻り値 】
なし

【 機　能 】
データ受信処理関数（usbf850_data_receive）を呼び出して Endpoint0 で受信したデータを取り込み，UART_MODE_INFO 構造体に書き込みます。またこの値を基に，転送速度やデータ長などの UART のモードを設定したあと，Endpoint0 用 NULL パケット送信処理関数（usbf850_sendnullEP0）を呼び出します。

usbf850_get_line_coding

【 概　要 】
Get Line Coding リクエスト処理

【C 言語記述形式】
void usbf850_get_line_coding(void)

【パラメータ】
なし

【 戻り値 】
なし

【 機　能 】
データ送信処理関数（usbf850_data_send）を呼び出して，UART_MODE_INFO 構造体の値を Endpoint0 から送信させます。

usbf850_set_control_line_state

【 概　要 】
Set Control Line State リクエスト処理

【C 言語記述形式】
void usbf850_set_control_line_state(void)

【パラメータ】
なし

【 戻り値 】
なし

【 機　能 】
Endpoint0 用 NULL パケット送信処理関数（usbf850_sendnullEP0）を呼び出します。

5. サンプル・アプリケーションの仕様

この章では，サンプル・ドライバに含まれるサンプル・アプリケーションについて説明します。

5.1 概　要

サンプル・アプリケーションは，USB コミュニケーション・デバイス・クラス用ドライバの利用の簡単な例として用意されたもので，サンプル・ドライバのメイン・ルーチンに組み込まれています。
このサンプル・アプリケーションは，USB ファンクション・コントローラで受信したデータの読み出し，読み出したデータの送信，の一連の処理を実行します。この処理の過程で，サンプル・ドライバの各種関数を利用します。

5.2 動　作

サンプル・アプリケーションでは，次のようなフローで一連の処理が行われます。

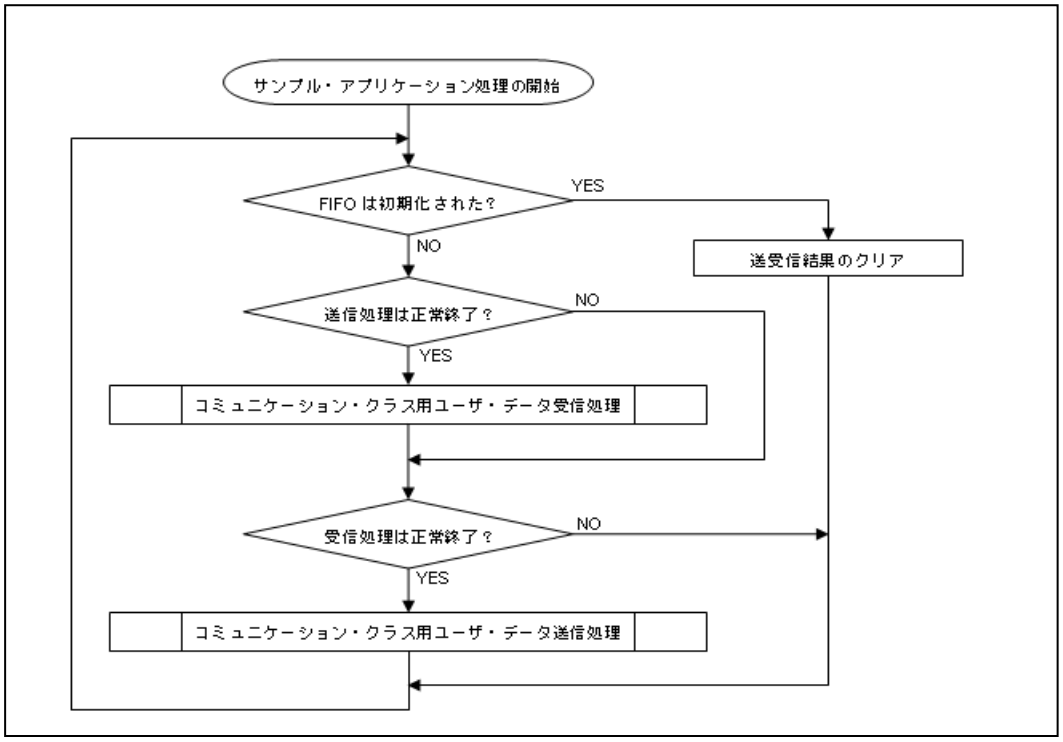


図 5−1　サンプル・アプリケーションの処理フロー

(1) ユーザ･データ用 FIFO 初期化確認

ユーザ・データ用の FIFO の状態通知処理関数（usbf850_get_bufinit_flg）を呼び出し、通常状態であれば送信処理結果の確認処理へ、初期化された状態であれば送受信結果のクリア処理（CDC 用ユーザ･データ送受信処理結果の 0 クリア）を行います。

(2) CDC 用ユーザ・データ送信処理結果確認

CDC 用ユーザ・データ送信処理の結果が正常終了(および、初期状態)であれば CDC 用ユーザ･データ受信処理を行い、異常終了状態であれば、受信処理結果の確認処理を行います。

(3) CDC 用ユーザ・データ受信処理

受信データを格納するバッファのサイズを指定し、CDC 用ユーザ・データ受信処理関数（usbf850_recv_buf）を呼び出します。

(4) CDC 用ユーザ・データ受信処理結果確認

CDC 用ユーザ・データの受信処理の結果が正常終了（および、初期状態）であれば CDC 用ユーザ・データ送信処理を行い、異常終了であれば、ユーザ・データ用 FIFO 初期化確認処理を行います。

(5) CDC 用ユーザ・データ送信処理

送信したいデータが格納されているバッファ･アドレス、送信するデータサイズを指定し、CDC 用ユーザ・データの送信処理関数（usbf850_send_buf）を呼び出します。

5.3　関数の利用

このサンプル・アプリケーションを含むソース・ファイル "main.c" では，サンプル・ドライバの関数を利用するために次のように記述しています。なお, 各関数の詳細は 4. 3　関数の仕様を参照してください。

```
1   #pragma ioreg                                                          ┐
2                                                                          │
3   #include "main.h"                                                     │
4   #include "usbf850_errno.h"                                            │
5   #include "usbf850.h"                                                  │
6   #include "usbf850_communication.h"                                    │
                                                                           ├  ・・・(1)
7       :                                                                  │
8       中略                                                               │
9       :                                                                  │
                                                                           │
10  #define USERBUF_SIZE   64      /* user buffer size */                  │
11  static UINT8   UserBuf[USERBUF_SIZE]; /* user buffer    */             │
12  extern UINT8   usbf850_rsuspd_flg;    /* resume/suspend flag */        ┘
        :
13      中略
        :
14  void main(void)
15  {
16      INT32 rcv_ret = 0;
17      INT32 snd_ret = 0;
18                                                                                ・・・(2)
19      cpu_init();
20
21      usbf850_init();/* initial setting of the USB Function */                  ・・・(3)
22
23      EI();
24
25      while (1) {
26          if (usbf850_get_bufinit_flg() != DEV_ERROR) {                         ・・・(4)
27              if (snd_ret >= 0) {
28                  rcv_ret = usbf850_recv_buf(&UserBuf[0], USERBUF_SIZE);        ・・・(5)
29              }
30              if (rcv_ret >= 0) {
31                  snd_ret = usbf850_send_buf(&UserBuf[0], rcv_ret);             ・・・(6)
32              }
33          }
34          else {                                                                ・・・(7)
35              snd_ret = 0;
36              rcv_ret = 0;
37          }
38
39          if (usbf850_rsuspd_flg == SUSPEND) {                                  ・・・(8)
40
41              DI();
42
43              halt();
44
45              usbf850_rsuspd_flg = RESUME;
46
47              EI();
48          }
49      }
50  }
```

図 5―2　サンプル・アプリケーションの記述（部分）

(1) 各種定義と宣言

サンプル･ドライバの関数で使用するため, "usbf850.h" と "usbf850_communication.h" の 2 つのヘッダ・ファイルをインクルードしています。また、ユーザ・データ用に 1 パケットのデータを処理するのに十分なサイズのユーザ・バッファを用意しています(USB の Full-Speed におけるバルク･エンドポイントの最大パケット・サイズは 64Byte です)。

(2) CPU 初期化処理

CPU 初期化処理関数（cpu_init）を呼び出します。

(3) USB ファンクション・コントローラ初期化処理

USB ファンクション・コントローラ初期化処理関数(usbf850_init)を呼び出します。

(4) ユーザ･データ用 FIFO 状態確認

ユーザ・データ用 FIFO の状態通知処理関数(usbf850_get_bufinit_flg)を呼び出し、FIFO の状態を確認します。

(5) ユーザ・データの受信処理

CDC 用ユーザ・データ受信処理関数(usbf850_recv_buf)を呼び出し、結果を保存します。

(6) ユーザ・データの送信処理

CDC 用ユーザ・データ送信処理関数(usbf850_send_buf)を呼び出し、結果を保存します。

(7) 送受信処理結果クリア処理

ユーザ・データ用 FIFO が初期化されると、(5)、(6)で保存されている送受信処理結果をクリア（0）します。

(8) サスペンド・レジューム処理

サスペンド発生を示すフラグ(usbf850_rsuspd_flg)の監視を行います。サスペンドが発生した場合、__halt()関数を呼び出し HALT 状態に遷移します。レジュームは外部割込みを以って行われ、サスペンド発生を示すフラグ(usbf850_rsuspd_flg)を RESUME 状態にして動作を再開します。

6. 開発環境

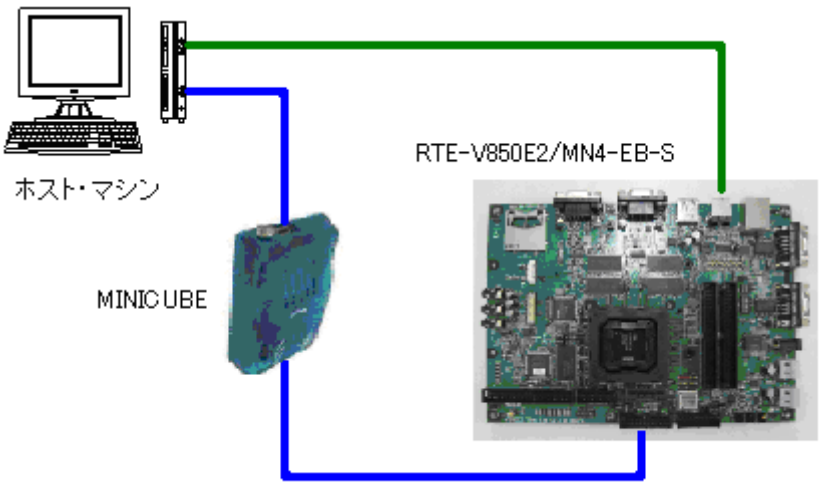
この章では，V850E2/MN4 向け USB コミュニケーション・デバイス・クラス用サンプル・ドライバを利用したアプリケーション・プログラムを開発する際の環境構築の例と，そこでのデバッグの手順について説明します。

6.1 開発環境

ここでは，ハードウエア・ツールとソフトウエア・ツールの製品構成例を示します。

6.1.1 システム構成

サンプル・ドライバを利用するシステムの構成を図 6－1 に示します。



（備考）RTE-V850E2/MN4-EB-S の外形やポートについては，「8. スタータ・キット概説」を参照してください。

図 6－1　開発環境のシステム構成

6.1.2 プログラム開発

サンプル・ドライバを利用したシステムを開発する際には，次のようなハードウエアとソフトウエアが必要です。

表 6-1　プログラム開発環境構成例

構成品		製品例	備　　考
ハードウエア	ホスト・マシン	－	PC/AT™互換機 （OS：Windows XPまたはWindows Vista®）
ソフトウエア	統合開発ツール	CubeSuite	V1.40
		Multi	V5.1.7D
		IAR Embedded Workbench	V3.71
	コンパイラ	CX850	V1.00
		CCV850	V5.1.7D
		ICCV850	V3.71.2

6.1.3 デバッグ

サンプル・ドライバを利用したシステムをデバッグする際には，次のようなハードウエアとソフトウエアが必要です。

表 6-2　デバッグ環境構成例

構成品		製品例	備　　考
ハードウエア	ホスト・マシン	－	PC/AT互換機 （OS：Windows XPまたはWindows Vista®））
	ターゲット	RTE-V850E2/MN4-EB-S	（株）マイダス・ラボ製
	USBケーブル	－	Bコネクタ - Aコネクタ
ソフトウエア	統合開発ツール・デバッガ	CubeSuite	V1.40
		Multi	V5.1.7D
		IAR Embedded Workbench	V3.71
ファイル	デバイス・ファイル	DF703512	V850E2/MN4用（CubeSuite/Multi/IAR Embedded Workbench用別）
	デバッグ・ポート用ホスト・ドライバ	－	(注12)
	プロジェクト関連ファイル	－	(注13)

（注 12）製品や入手方法については弊社までお問い合わせください。

（注 13）CubeSuite / Multi / IAR Embedded Workbench で構築した場合のファイルがサンプル・ドライバに同梱されています。

6.2 CubeSuite環境設定

ここでは，「6. 1　開発環境」に示した製品構成の中で，CubeSuite を用いた開発やデバッグを行うための準備について説明します。Multi を用いた開発やデバッグを行う場合には「6. 4　Multi 環境設定」を、IAR Embedded Workbench を用いた開発やデバッグを行う場合には「6. 6　IAR Embedded Workbench 環境設定」参照してください。

6.2.1 ホスト環境整備

ホスト・マシン上に専用のワークスペースを作成します。

(1) CubeSuite 統合開発ツールのインストール

CubeSuite をインストールします。詳細は CubeSuite のユーザーズマニュアルを参照してください。

(2) ドライバ類の展開

サンプル・ドライバの提供ファイル一式を，フォルダ構成を変えずに任意のディレクトリに格納します。

また，デバッグ・ポート用ホスト・ドライバを任意のディレクトリに格納します。

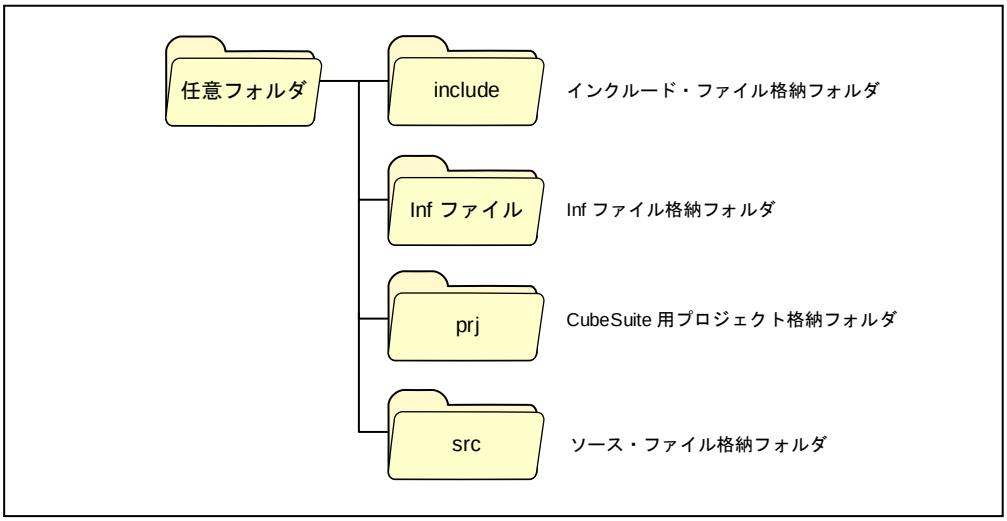


図 6−2　サンプル・ドライバのフォルダ構成（CubeSuite 版）

(3) デバイス・ファイルのインストール

CubeSuite 用 V850E2/MN4 用のデバイス・ファイルを，CubeSuite インストールフォルダにコピーします。

例）　C:¥Program Files¥NEC Electronics CubeSuite¥CubeSuite¥Device_Custom

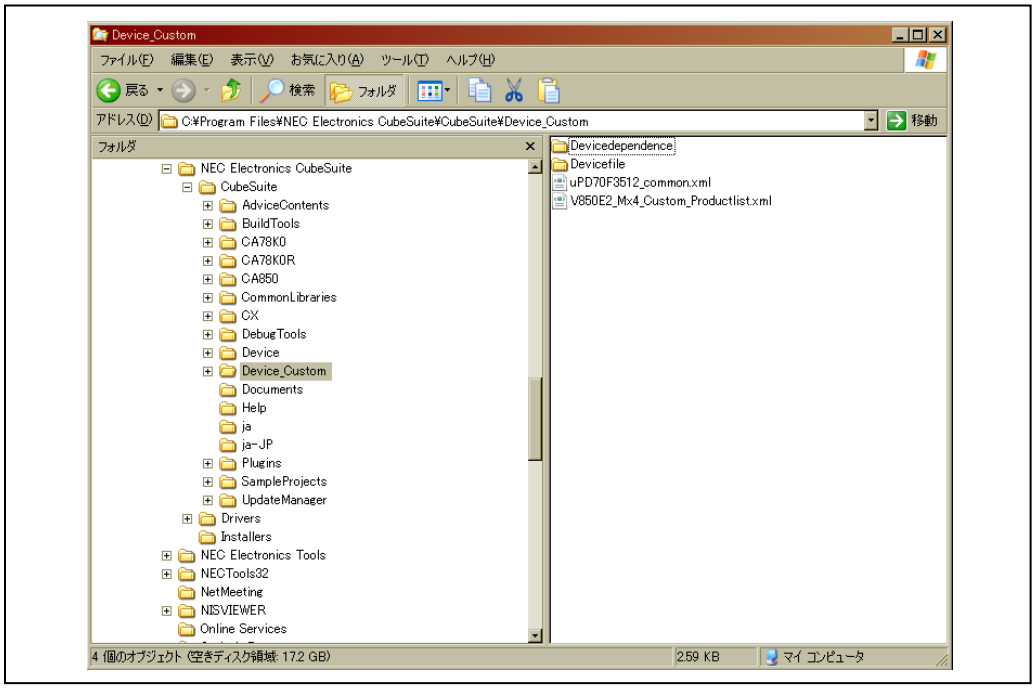


図 6−3　デバイス・ファイルのコピー先の例

(4) ワークスペースの設定

ここではサンプル・ドライバに同梱のプロジェクト関連ファイルを使用する場合の手順を示します。

<1>　CubeSuite を起動し，「ファイル」メニューから「ファイルを開く」を選択します。

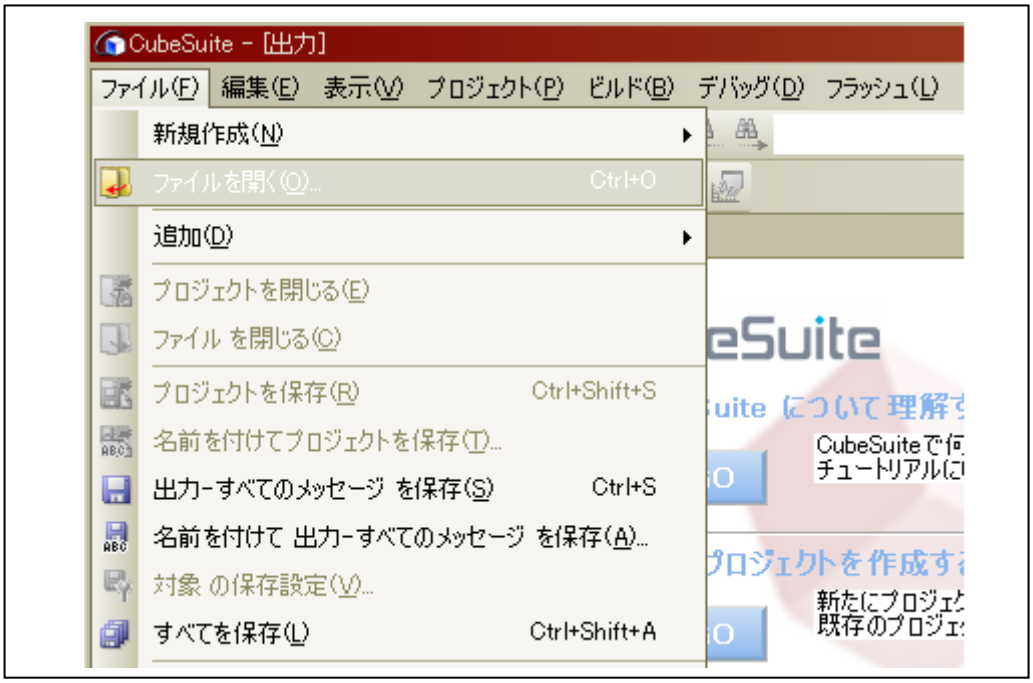


図 6−4　CubeSuite メニュー選択

<2>　「ファイルを開く」ダイアログが開きます。サンプル・ドライバを格納したディレクトリの「prj」フォルダにある CubeSuite 用プロジェクト・ファイルを指定します。

図 6−5　CubeSuite プロジェクト・ファイル選択

(5) ビルド・ツールの設定

ここではビルド・ツールとして使用する CX850 のバージョン選択と，デバッグ・ツールとして V850E2M　MINICUBE を使用する場合の手順を示します。

<1>　CubeSuite の「プロジェクト・ツリー」から「CX(ビルド・ツール)」を選択し，プロパティを表示します。

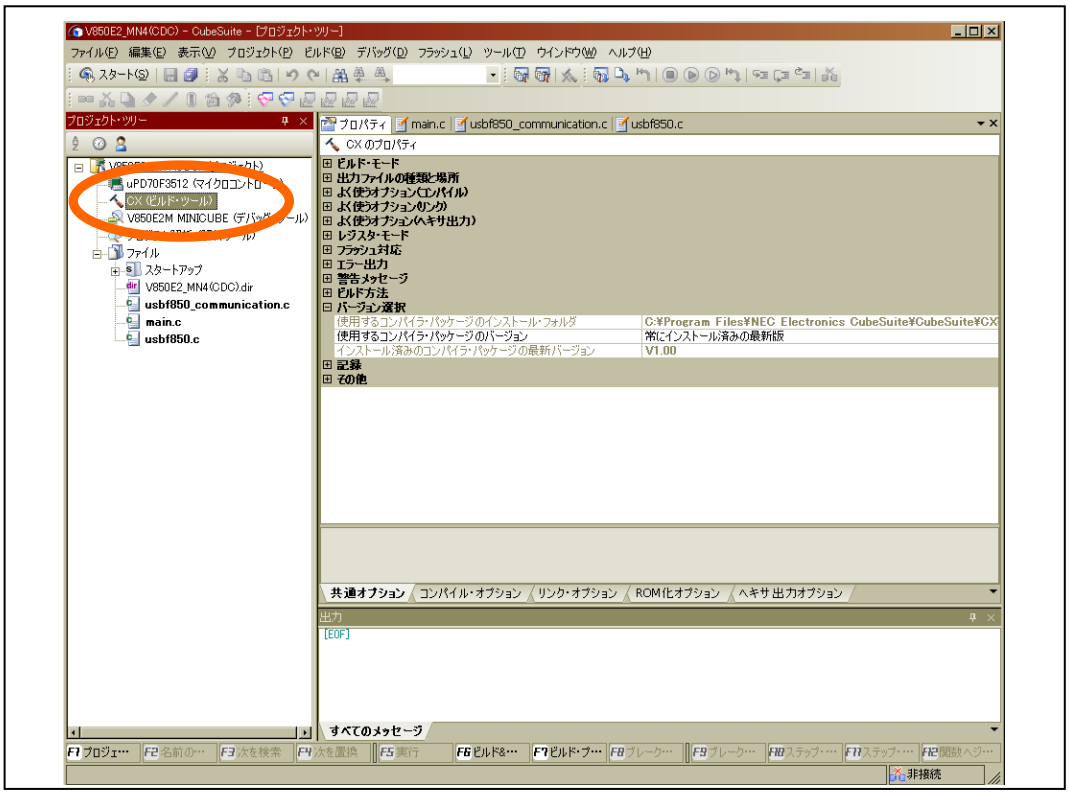


図 6−6　ビルド・ツール選択

<2>　「バージョン選択」の項目を選択し，「使用するコンパイラ・パッケージのバージョン」の項を「常にインストール済みの最新版」に設定します。

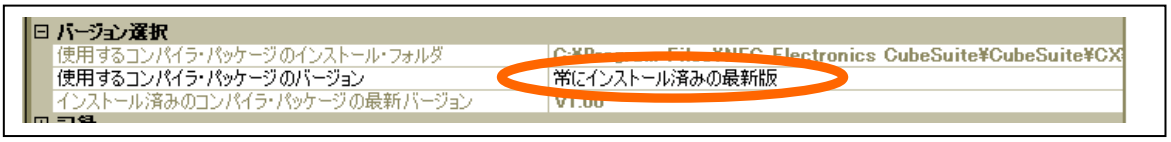


図 6−7　コンパイラ・パッケージ設定

<3>　プロジェクト・ツリーより「V850E2M　MINICUBE(デバッグ・ツール)」を選択し，右クリックメニューから「使用するデバッグ・ツール」→「V850E2M MINICUBE」を選択します。

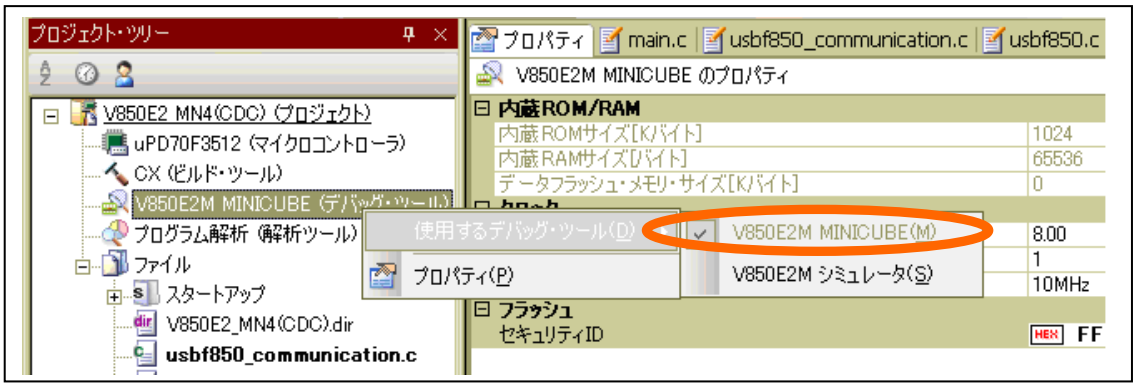


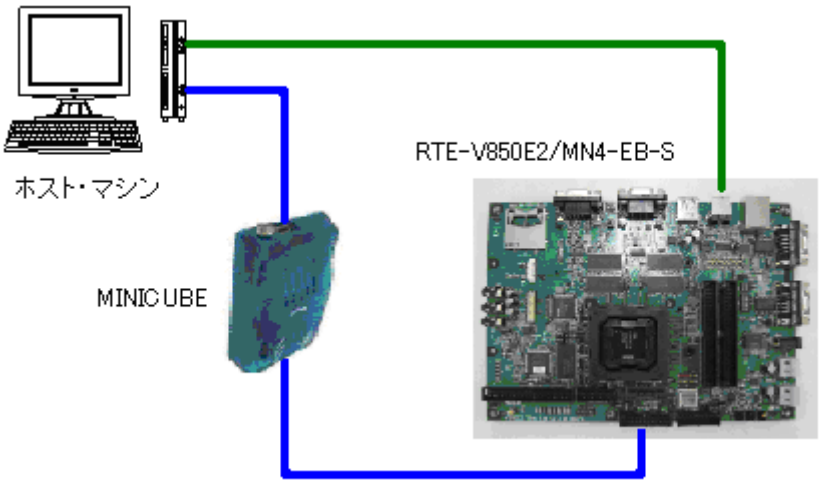
図 6−8　デバッグ・ツール選択

6.2.2　ターゲット環境整備

デバッグに使用するターゲット・デバイスを接続します。接続方法は CubeSuite/Multi/IAR Embedded Workbench 共に同じです。

(1) デバッグ・ポートの接続

RTE-V850E2/MN4-EB-S とホスト・マシンを接続します。RTE-V850E2/MN4-EB-S とホスト・マシンをデバッグ用に MINICUBE で接続します。また RTE-V850E2/MN4-EB-S の USB B タイプコネクタとホスト・マシンの USB コネクタを CDC 用に接続します。



（備考）RTE-V850E2/MN4-EB-S の外形やポートについては，「8. スタータ・キット概説」を参照してください。

図 6－9　RTE-V850E2/MN4-EB-S の接続

(2) ホスト・ドライバのインストール

ここではサンプル・ドライバに同梱の仮想 COM ポート用ホスト・ドライバを使用する場合の手順を示します。

<1> RTE-V850E2/MN4-EB-S の接続がホスト・マシンに認識されると，「新しいハードウエアが見つかりました」というメッセージが表示されたあと，新しいハードウエアの検出ウィザードが起動します。

<2> 「新しいハードウエアの検出ウィザード」ダイアログが開きます。「いいえ、今回は接続しません」を選択して「次へ」ボタンを押下します。

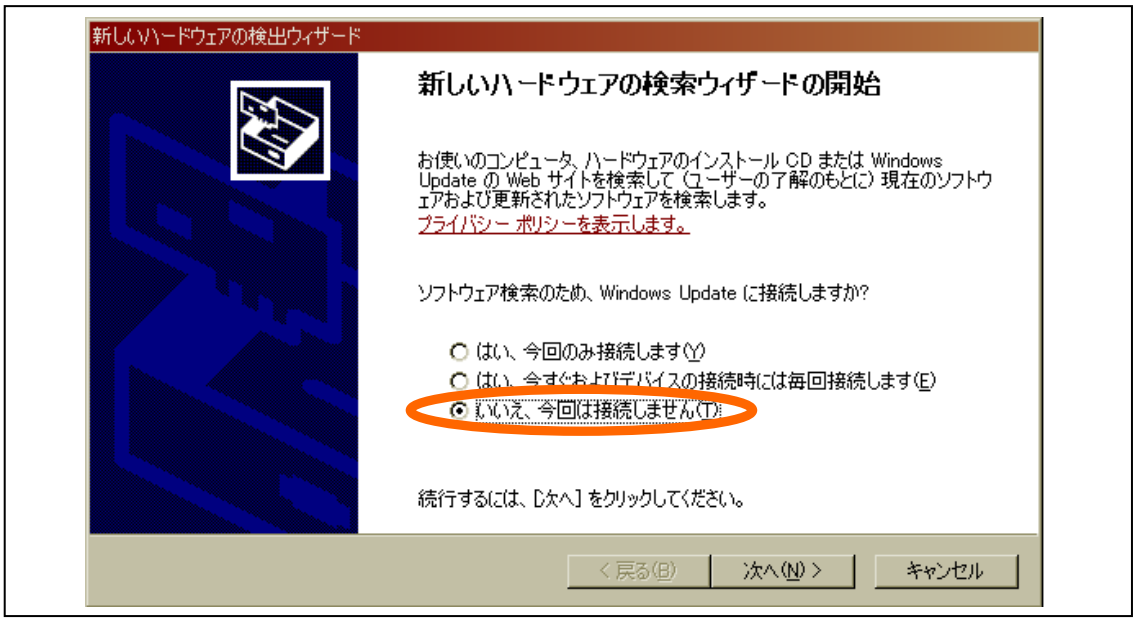


図 6－10　新しいハードウエアの検出ウィザード(1)

<3> 次の画面が表示されます。「一覧または特定の場所からインストールする（詳細）」を選択して「次へ」ボタンを押下します。

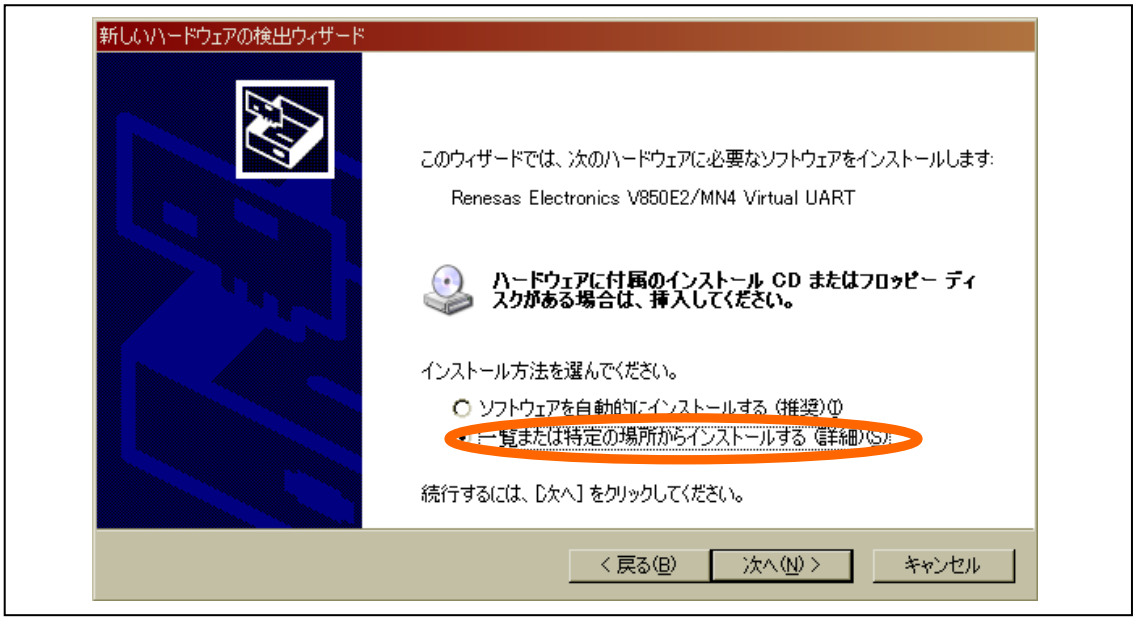


図 6－11　新しいハードウエアの検出ウィザード(2)

<4>　次の画面が表示されます。「検索しないで、インストールするドライバを選択する」を選択して「次へ」ボタンを押下します。

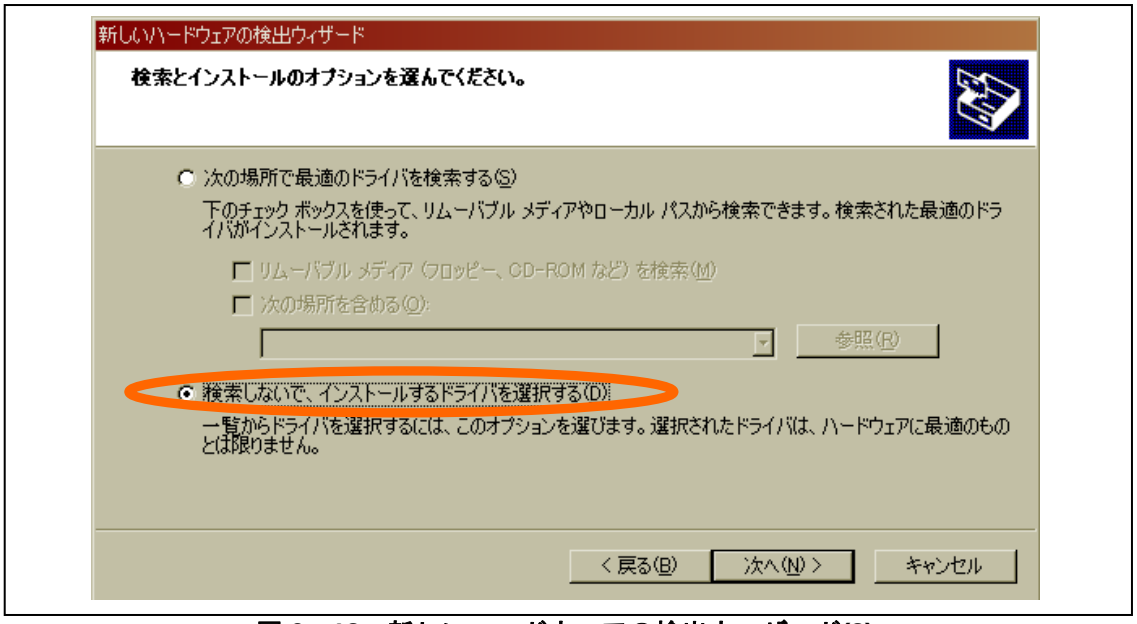


図 6－12　新しいハードウエアの検出ウィザード(3)

<5>　次の画面が表示されます。「ディスク使用」ボタンを押下します。

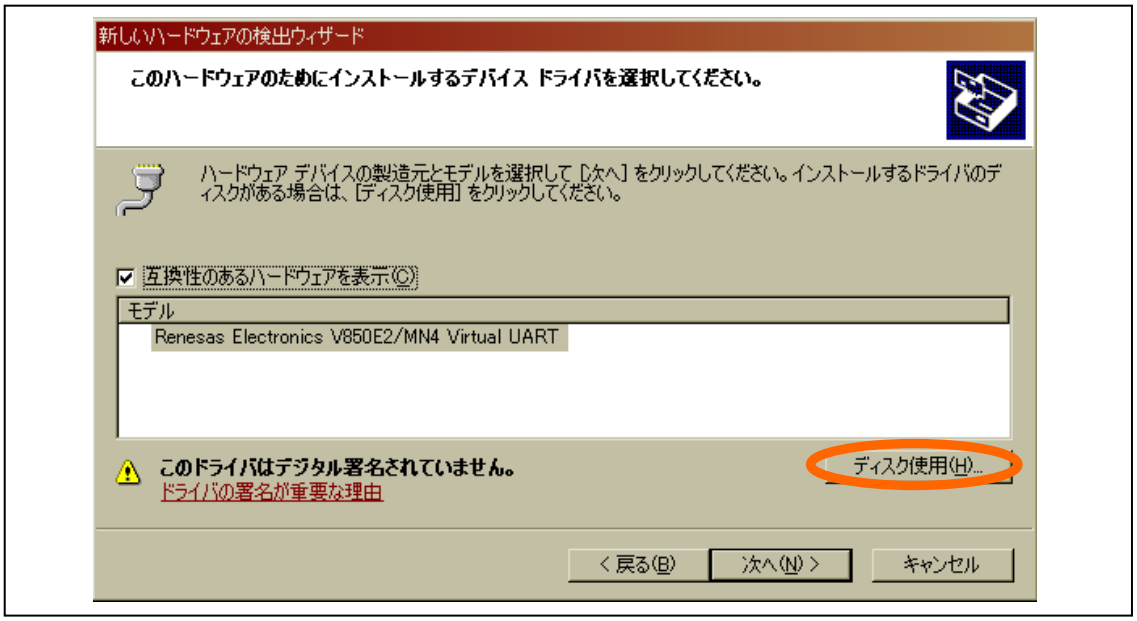


図 6－13　新しいハードウエアの検出ウィザード(4)

<6>　「フロッピーディスクからインストール」ダイアログが開きます。「参照」ボタンを押下して，サンプル・ドライバを格納したディレクトリの「Inf ファイル」フォルダを表示させます。

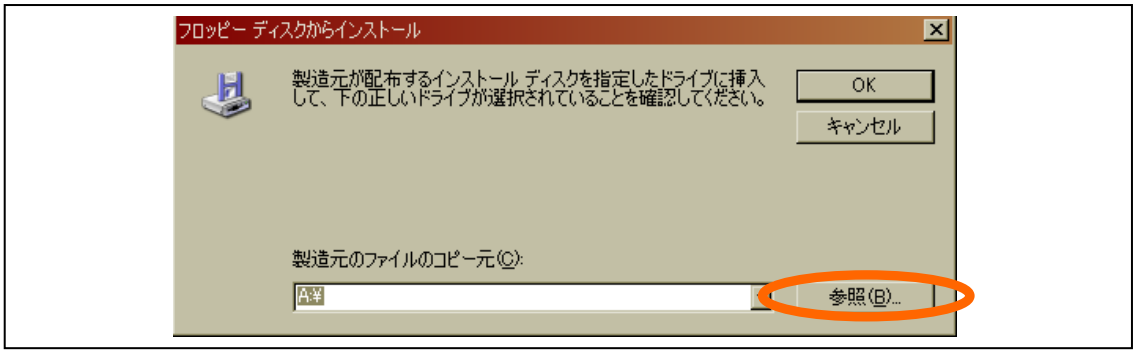


図 6－14　Inf ファイルの参照

<7>　ホスト・マシンで使用している OS にあわせ「XP」「VISTA」「Win7」フォルダの中の INF ファイルを選択し，「開く」ボタンを押下します。

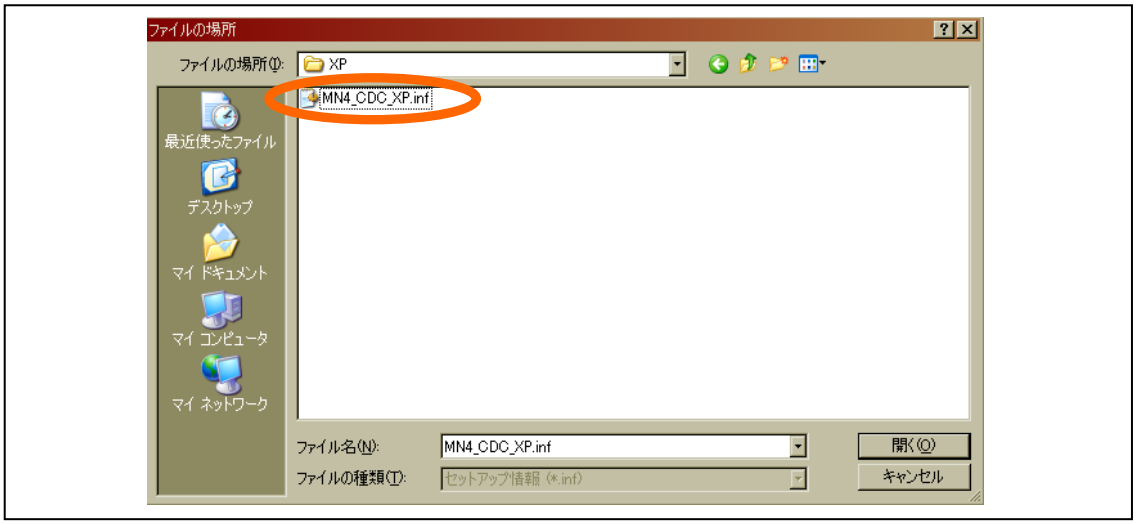


図 6－15　Inf ファイル選択

<8>　「フロッピーディスクからインストール」ダイアログに戻ります。「製造元のファイルのコピー元」欄が正しいことを確認し，「OK」ボタンを押下します。

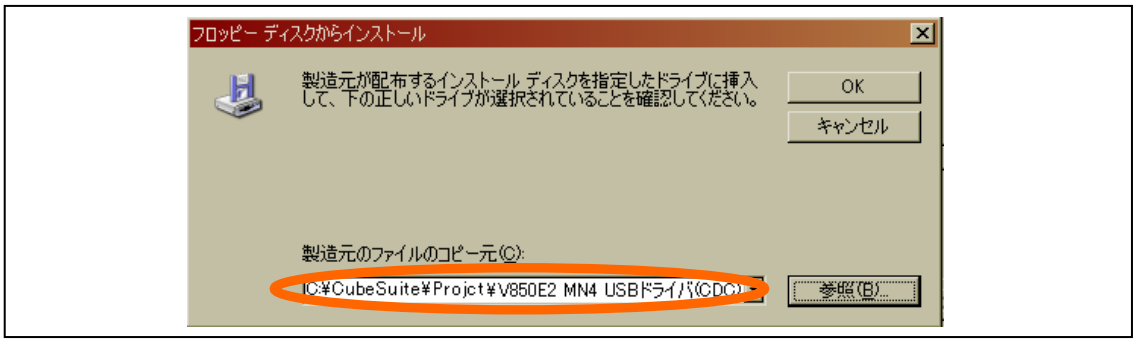


図 6－16　Inf ファイルインストール

<9>　「新しいハードウエアの検出ウィザード」画面に戻ります。「モデル」欄で「Renesas Electronics V850E2/MN4 Virtual UART」を選択して「次へ」ボタンを押下します。

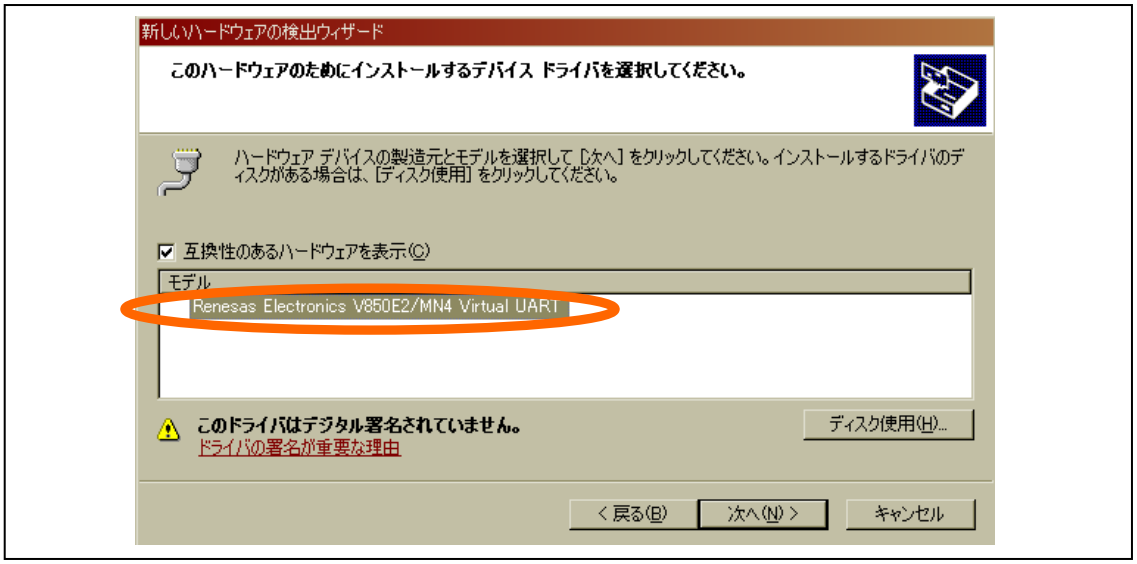


図 6－17　インストールデバイスドライバ選択

<10>　ドライバのインストールが開始されます。

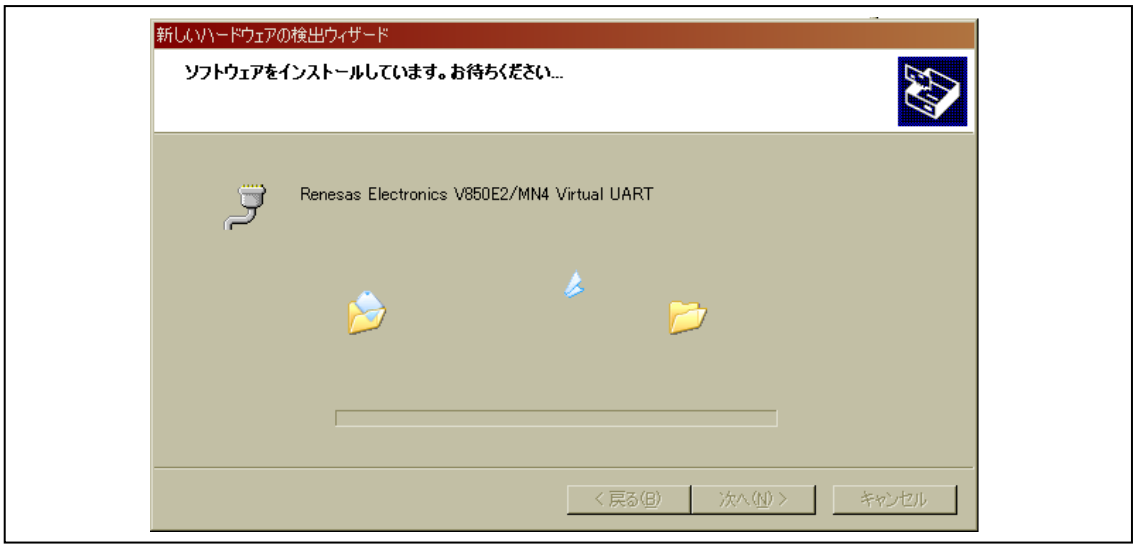


図 6－18　　ドライバインストール中(1)

<11> 「ハードウエアのインストール」ダイアログが表示されます。「続行」を押下します。

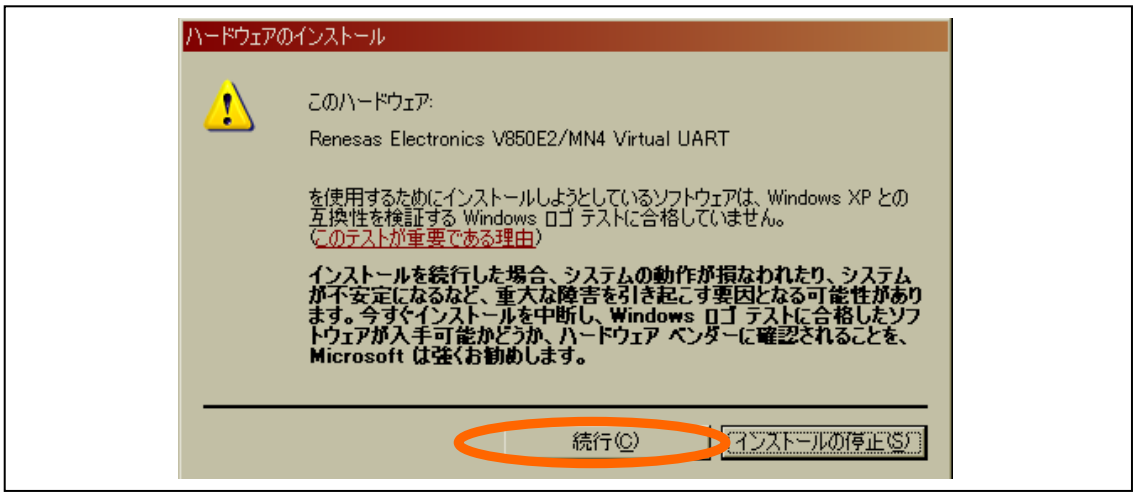


図 6－19　ドライバ互換性確認ダイアログ

<12> ドライバがインストールされます。環境により，時間がかかる場合があります。

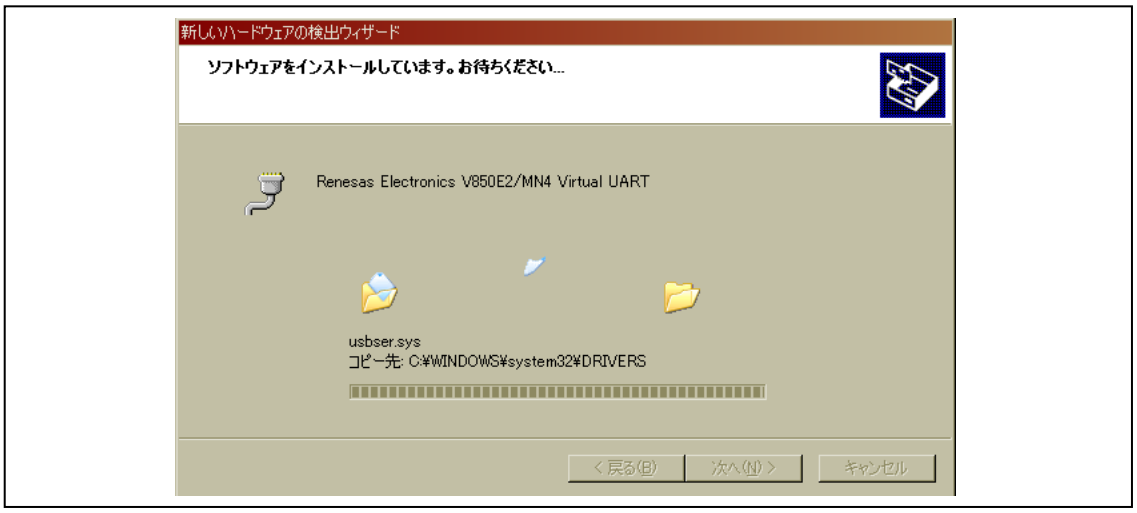


図 6－20　ドライバインストール中(2)

<13> 次の画面が表示されます。「完了」ボタンを押下します。

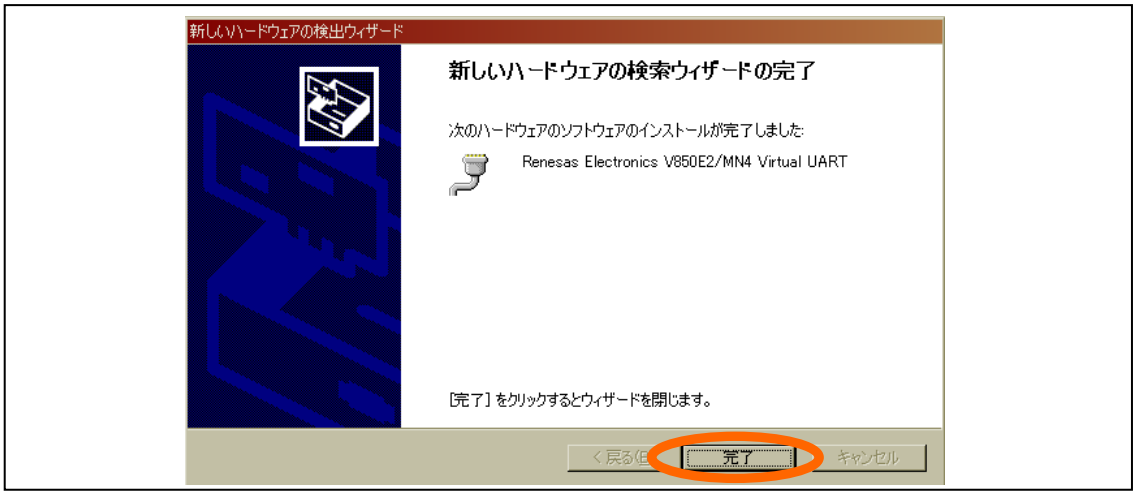


図 6－21　ドライバインストールの完了

(3) デバイス割り当ての確認

Windows のデバイスマネージャを開きます。デバイスの一覧表示の「ポート」のツリーを展開し，「Renesas Electronics V850E2/MN4 Virtual UART」が表示されていること，また割り当てられた COM ポート番号を確認します。

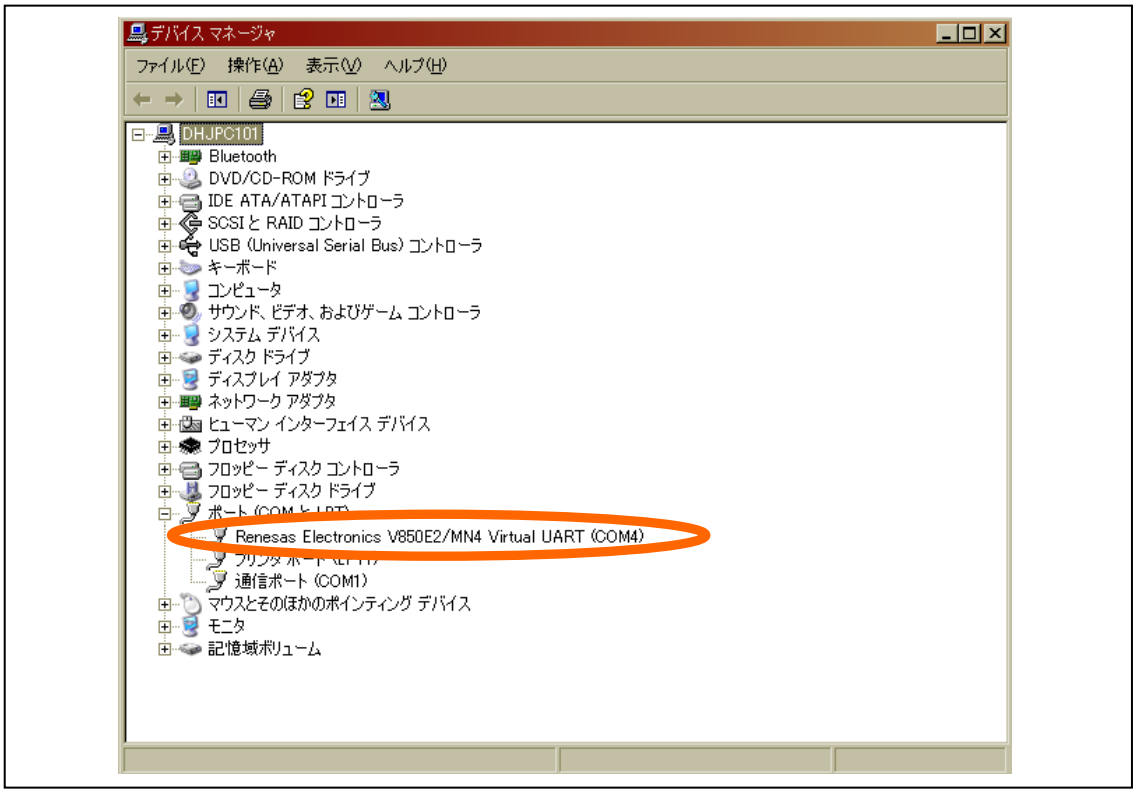


図 6－22　COM ポート確認

備考　デバイス名やポート番号は任意のものに変更できます。詳細は 7.2 カスタマイズを参照してください。

6.3 CubeSuite環境デバッグ

ここでは，「6. 2　CubeSuite 環境設定」に示したワークスペースで開発したアプリケーション・プログラムのデバッグ手順について説明します。

6.3.1 ロード・モジュール生成

ターゲット・デバイスにプログラムを書き込むには，C 言語やアセンブリ言語で記述されたファイルを C コンパイラなどで変換してロード・モジュールを生成します。
CubeSuite では，「ビルド」メニューから「ビルド・プロジェクト」を選択すると，ロード・モジュールが生成されます。

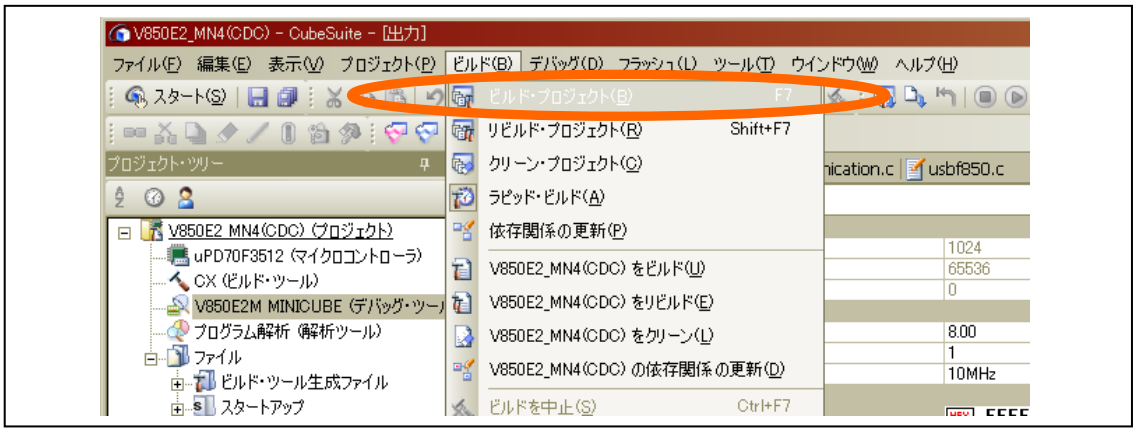


図 6－23　ビルド・プロジェクト

6.3.2　ロードと実行

生成したロード・モジュールをターゲットに書き込んで（ロード）実行させます。

(1) ロード・モジュールの書き込み

ここでは CubeSuite を介して RTE-V850E2/MN4-EB-S にロード・モジュールを書き込む手順を示します。

<1>　「デバッグ」メニューから「デバッグ・ツールへダウンロード」を選択してデバッガを起動します。

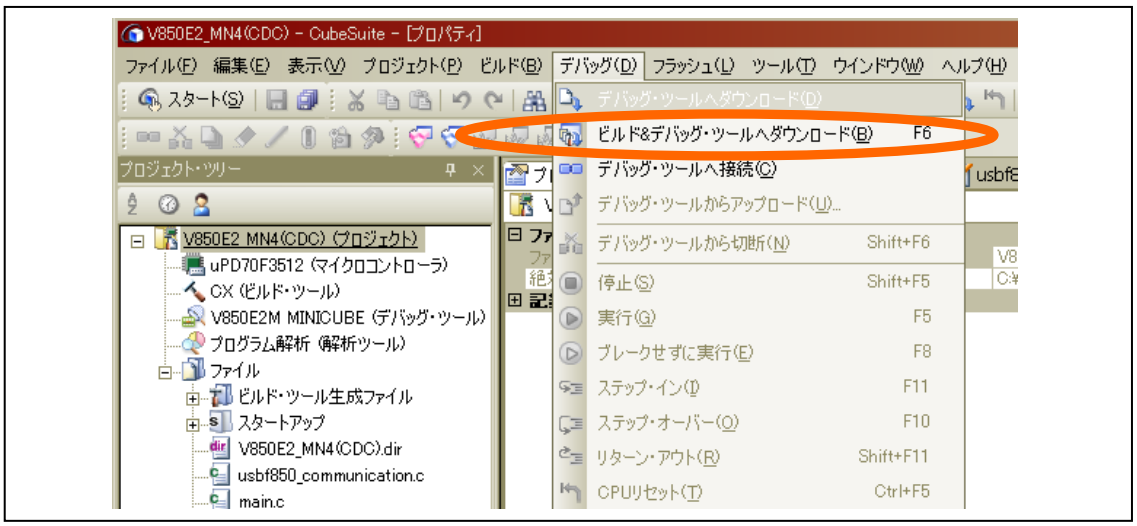


図 6－24　デバッガ起動

<2>　デバッグ・ツールを介して，ロード・モジュールのダウンロードが開始されます。

図 6－25　ダウンロード実行

(2) プログラムの実行

CubeSuite の  ボタンを押下します。または「デバッグ」メニューから「実行」を選択します。

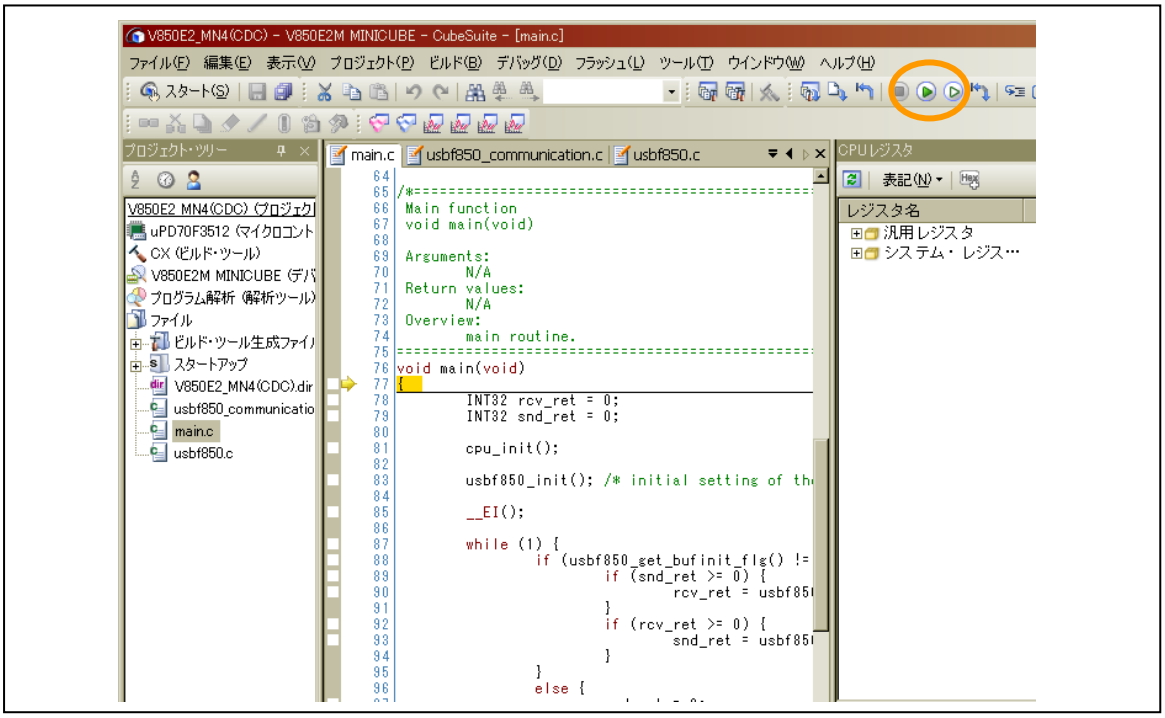


図 6－26　実行開始

6.4 Multi環境設定

ここでは，「6. 1　開発環境」に示した製品構成の中で，Multi を用いた開発やデバッグを行うための準備について説明します。

6.4.1 ホスト環境整備

ホスト・マシン上に専用のワークスペースを作成します。

(1) Multi 統合開発ツールのインストール

Multi をインストールします。詳細は GHS のユーザーズマニュアルを参照してください。

(2) ドライバ類の展開

サンプル・ドライバの提供ファイル一式を，フォルダ構成を変えずに任意のディレクトリに格納します。

また，デバッグ・ポート用ホスト・ドライバを任意のディレクトリに格納します。

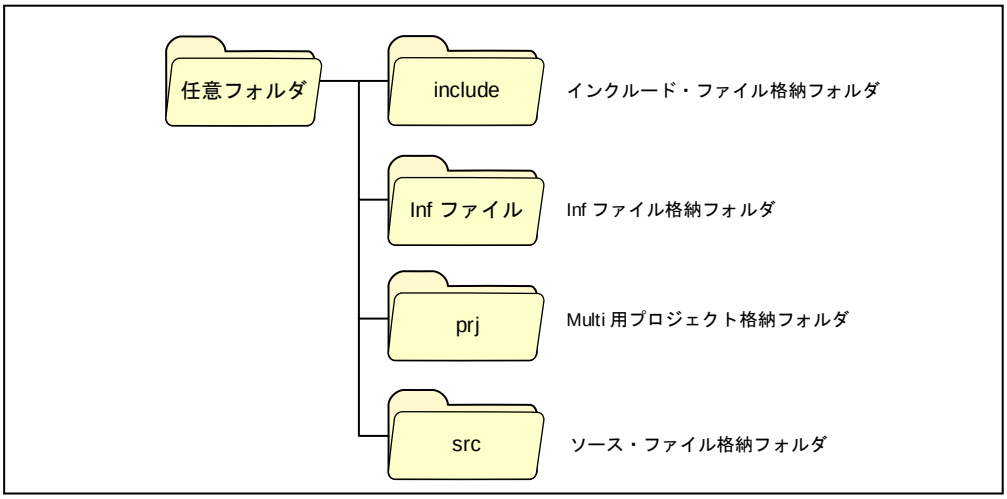


図 6－27　サンプル・ドライバのフォルダ構成（Multi 版）

(3) デバイス・ファイルのインストール

Multi 用 V850E2/MN4 用のデバイス・ファイルを，Multi インストールフォルダにコピーします。

例）　C:¥Green¥V800.V517D¥devicefile

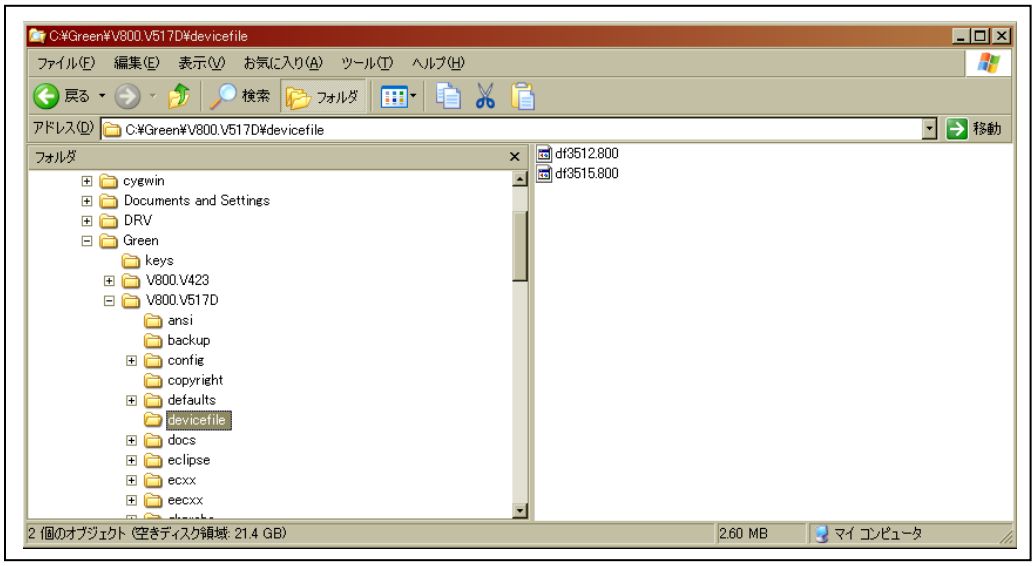


図 6－28　デバイス・ファイルのコピー先の例

(4) Multi の起動

エクスプローラから，サンプル・ドライバに同梱されている「V850E2_MN4(CDC)_GHS.gpj」の Multi Project File を選択し，Multi を起動します。

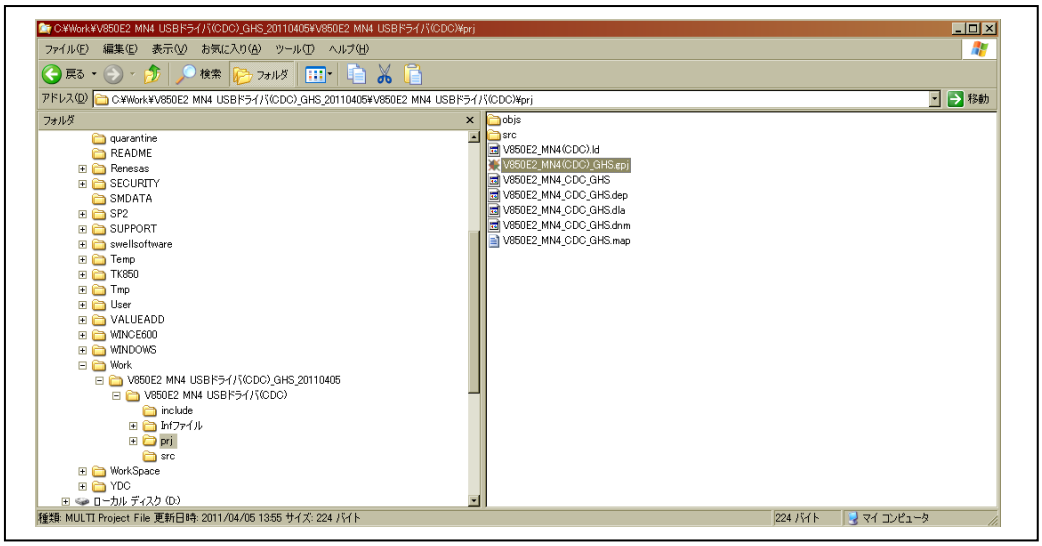


図 6－29　Multi Project File の選択

(5) デバッグ・ツールの設定

ここでは，デバッグ・ツールとして MINICUBE を使用する場合の手順を示します。

<1>　Multi の「Connect」から「Connect」を選択し，Connection Chooser を表示します。

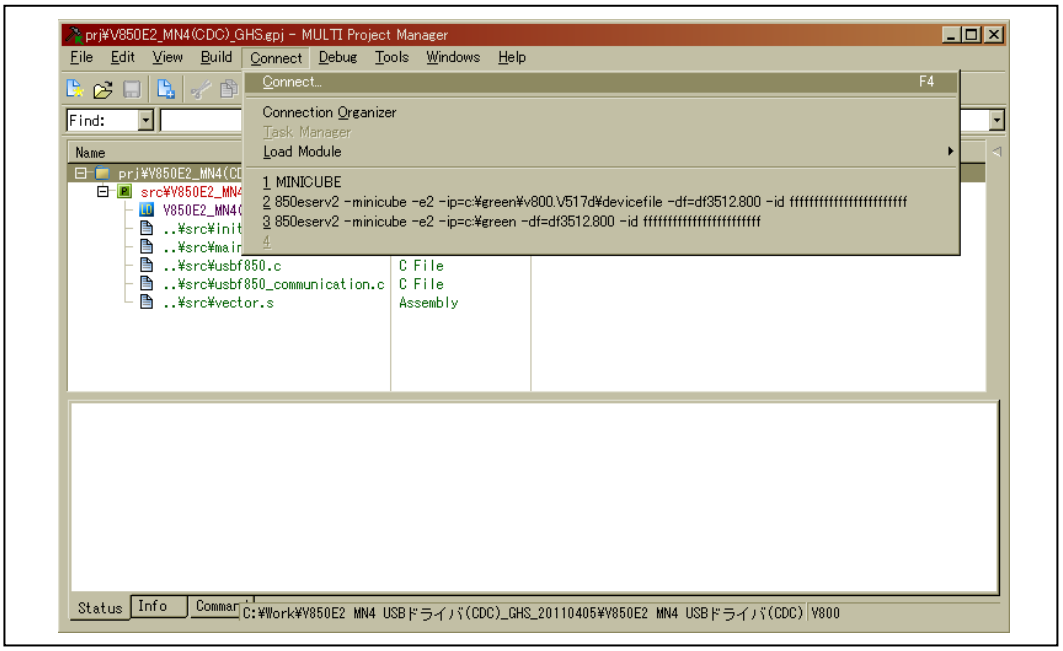


図 6－30　Connection Chooser の起動

<2>　Connection Chooser ダイアログから，「Create a new Connection Method」アイコンを選択します。

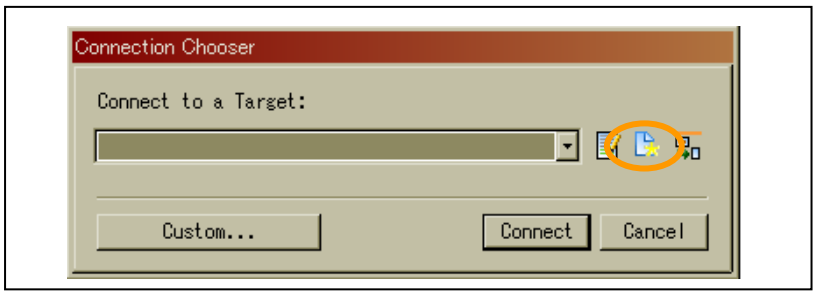


図 6－31　Create a new Connection Method の選択

<3>　Create New Connection Method ダイアログで Name に任意の設定名を，Type に「Custom」を選択して「Create…」ボタンを押下し，MINICUBE 接続設定を作成します。ここでは Name に「MINICUBE」を設定した例を示します。

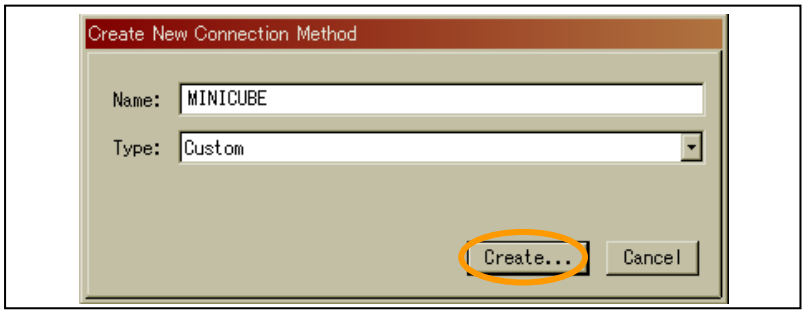


図 6－32　Create New Connection Method の作成

<4>　Connection Editor が起動するので，「Server」及び「Arguments」に以下の設定を記述して OK ボタンを押下します。

Server　　：850eserv2

Arguments：-minicube -e2 -ip=c:¥green¥v800.V517d¥devicefile -df=df3512.800 -id ffffffffffffffffffffffff（注 10）

（注 10）”f”を 24 個

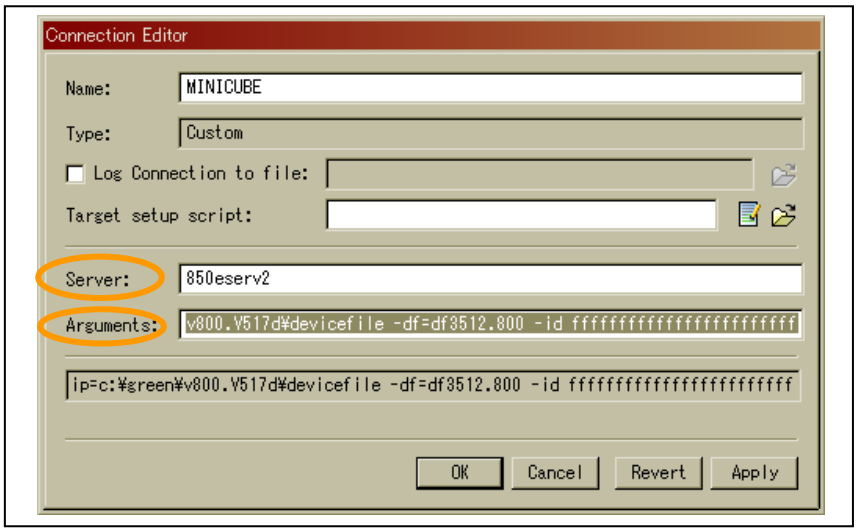


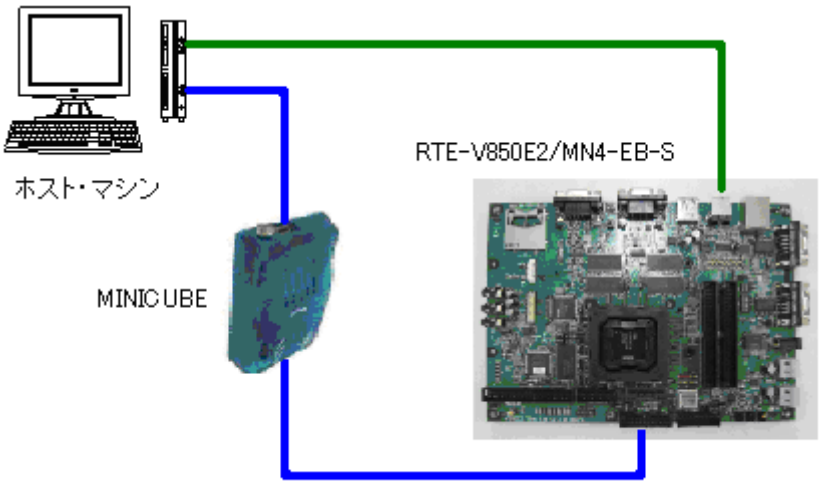
図 6－33　Connection Editor の設定

6.4.2　ターゲット環境整備

デバッグに使用するターゲット・デバイスを接続します。接続方法は CubeSuite/Multi/IAR Embedded Workbench 共に同じです。

(1) デバッグ・ポートの接続

RTE-V850E2/MN4-EB-S とホスト・マシンを接続します。RTE-V850E2/MN4-EB-S とホスト・マシンをデバッグ用に MINICUBE で接続します。また RTE-V850E2/MN4-EB-S の USB B タイプコネクタとホスト・マシンの USB コネクタを CDC 用に接続します。



（備考）RTE-V850E2/MN4-EB-S の外形やポートについては，「8. スタータ・キット概説」を参照してください。

図 6－34　RTE-V850E2/MN4-EB-S の接続

(2) ホスト・ドライバのインストール

ここではサンプル・ドライバに同梱の仮想 COM ポート用ホスト・ドライバを使用する場合の手順を示します。

<1> RTE-V850E2/MN4-EB-S の接続がホスト・マシンに認識されると，「新しいハードウエアが見つかりました」というメッセージが表示されたあと，新しいハードウエアの検出ウィザードが起動します。

<2> 「新しいハードウエアの検出ウィザード」ダイアログが開きます。「いいえ、今回は接続しません」を選択して「次へ」ボタンを押下します。

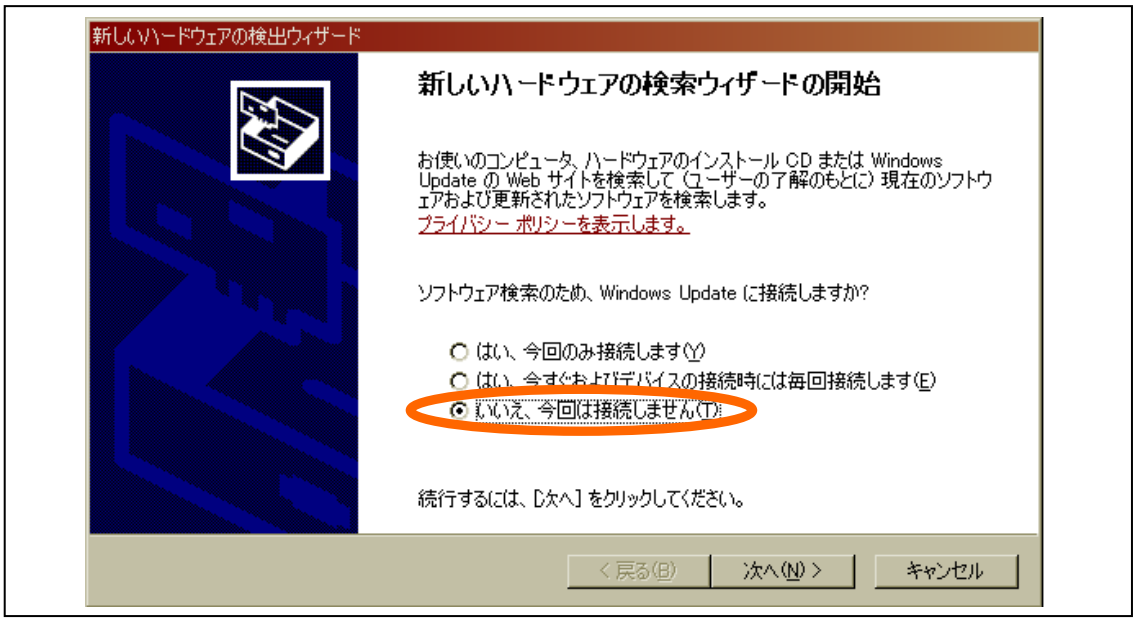


図 6－35　新しいハードウエアの検出ウィザード(1)

<3> 次の画面が表示されます。「一覧または特定の場所からインストールする（詳細）」を選択して「次へ」ボタンを押下します。

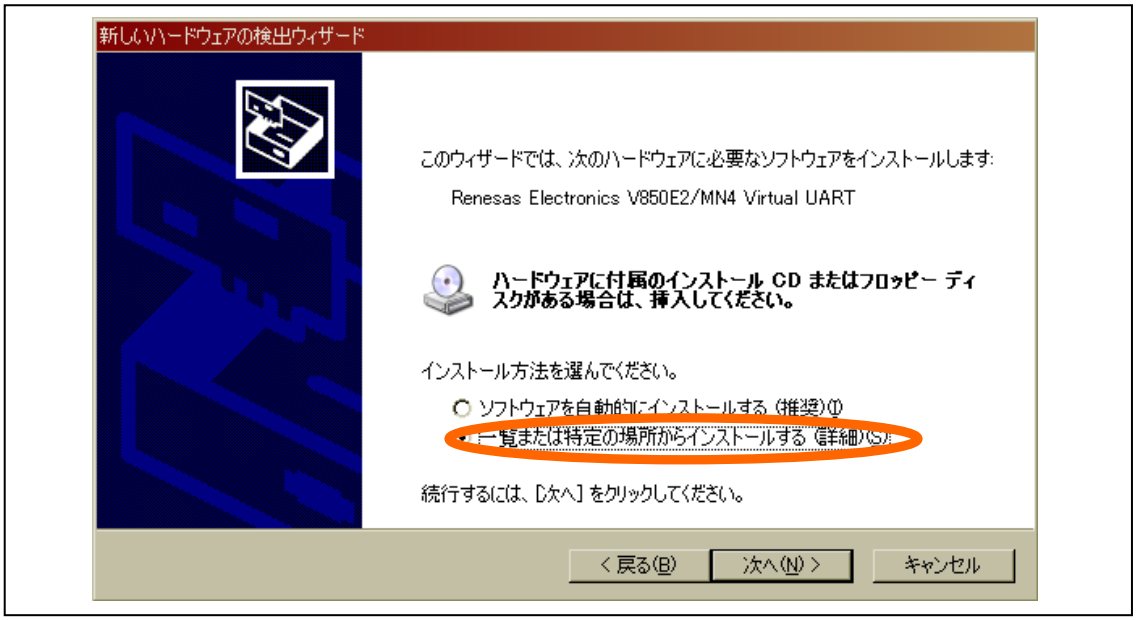


図 6－36　新しいハードウエアの検出ウィザード(2)

<4>　次の画面が表示されます。「検索しないで、インストールするドライバを選択する」を選択して「次へ」ボタンを押下します。

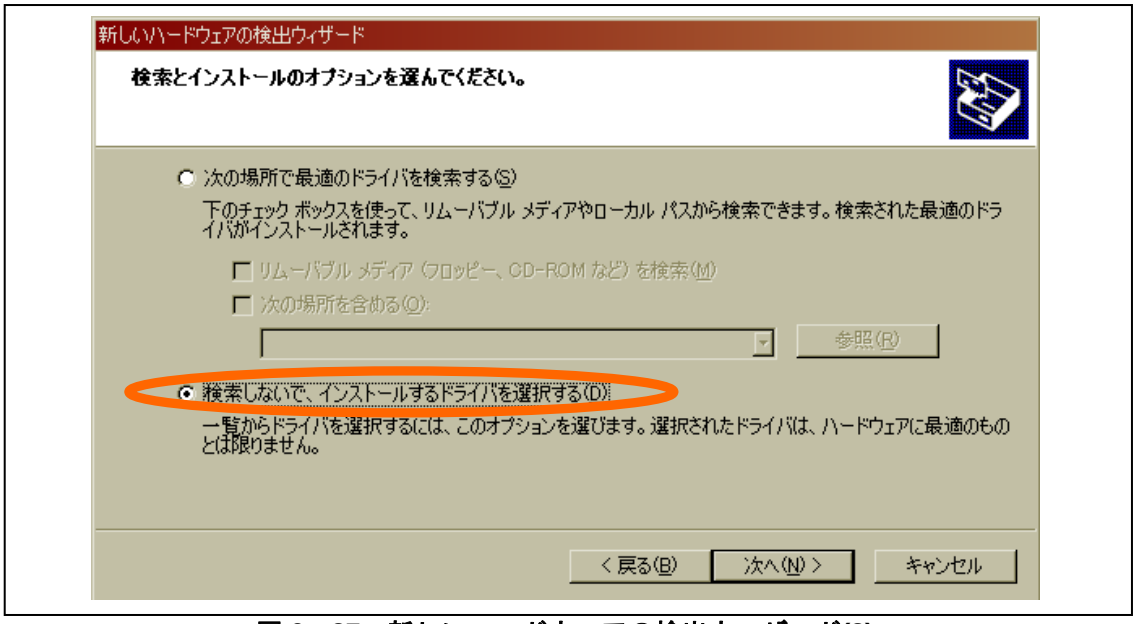


図 6－37　新しいハードウエアの検出ウィザード(3)

<5>　次の画面が表示されます。「ディスク使用」ボタンを押下します。

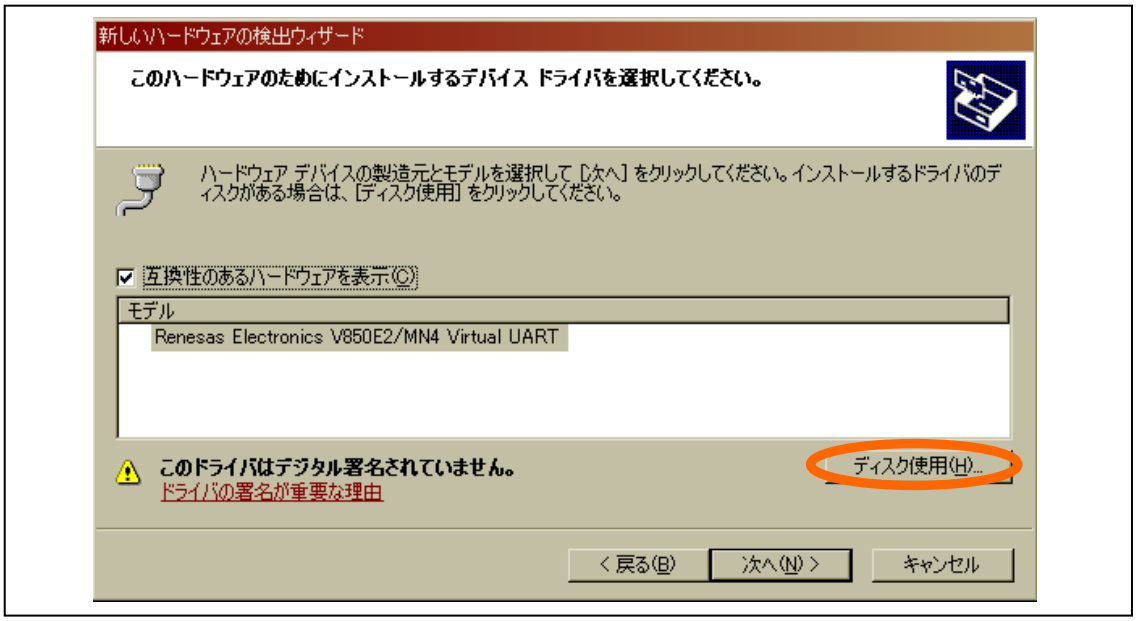


図 6－38　新しいハードウエアの検出ウィザード(4)

<6>　「フロッピーディスクからインストール」ダイアログが開きます。「参照」ボタンを押下して，サンプル・ドライバを格納したディレクトリの「Inf ファイル」フォルダを表示させます。

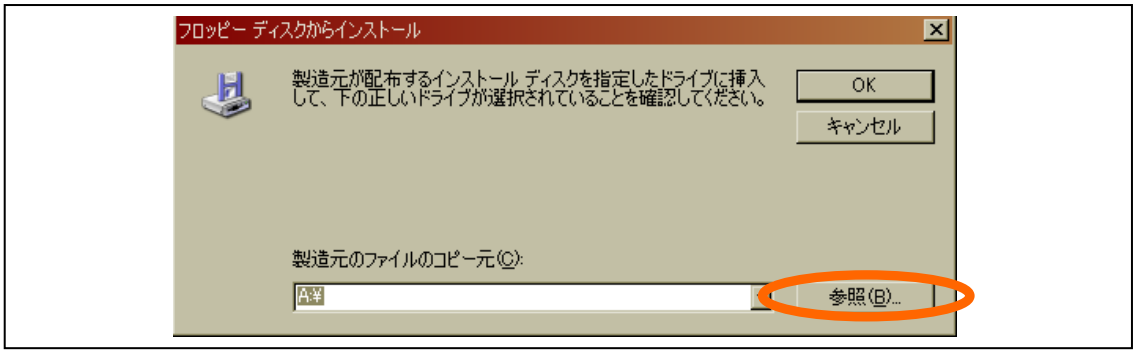


図 6－39　Inf ファイルの参照

<7>　ホスト・マシンで使用している OS にあわせ「XP」「VISTA」「Win7」フォルダの中の INF ファイルを選択し，「開く」ボタンを押下します。

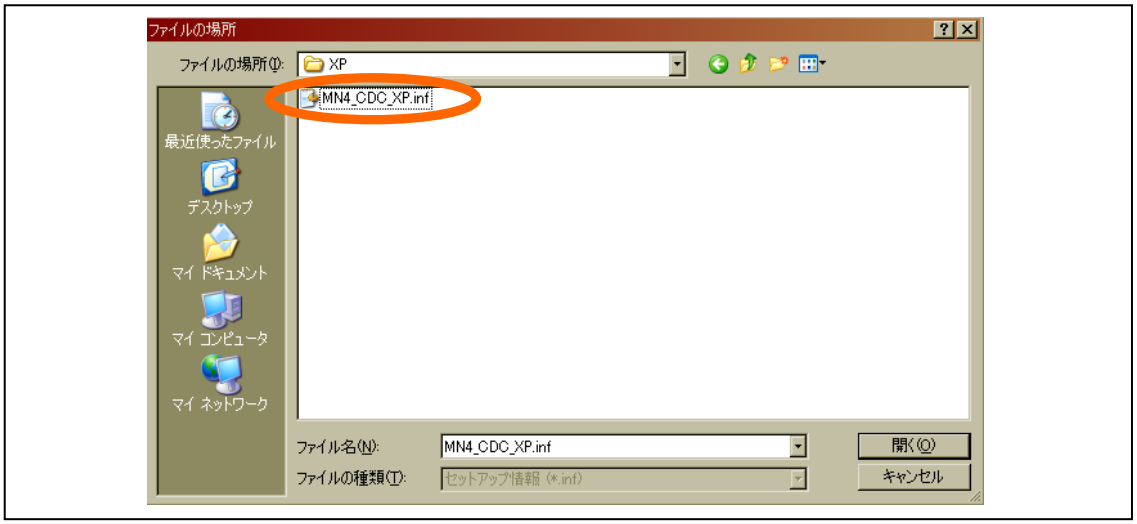


図 6－40　Inf ファイル選択

<8>　「フロッピーディスクからインストール」ダイアログに戻ります。「製造元のファイルのコピー元」欄が正しいことを確認し，「OK」ボタンを押下します。

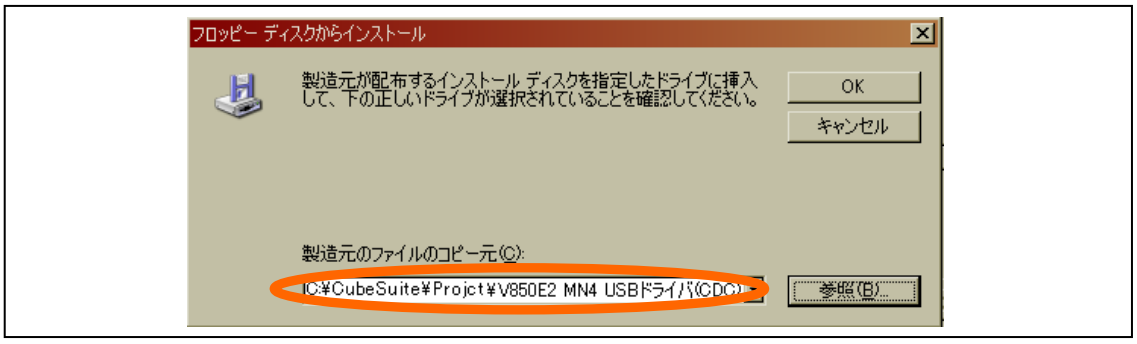


図 6－41　Inf ファイルインストール

<9>　「新しいハードウエアの検出ウィザード」画面に戻ります。「モデル」欄で「Renesas Electronics V850E2/MN4 Virtual UART」を選択して「次へ」ボタンを押下します。

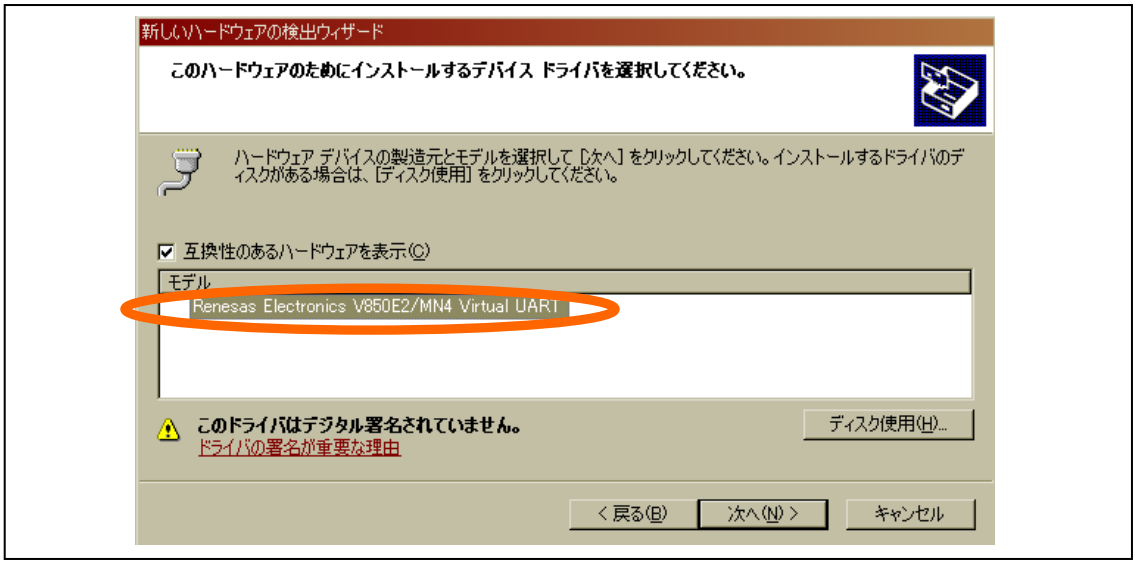


図 6－42　インストールデバイスドライバ選択

<10>　ドライバのインストールが開始されます。

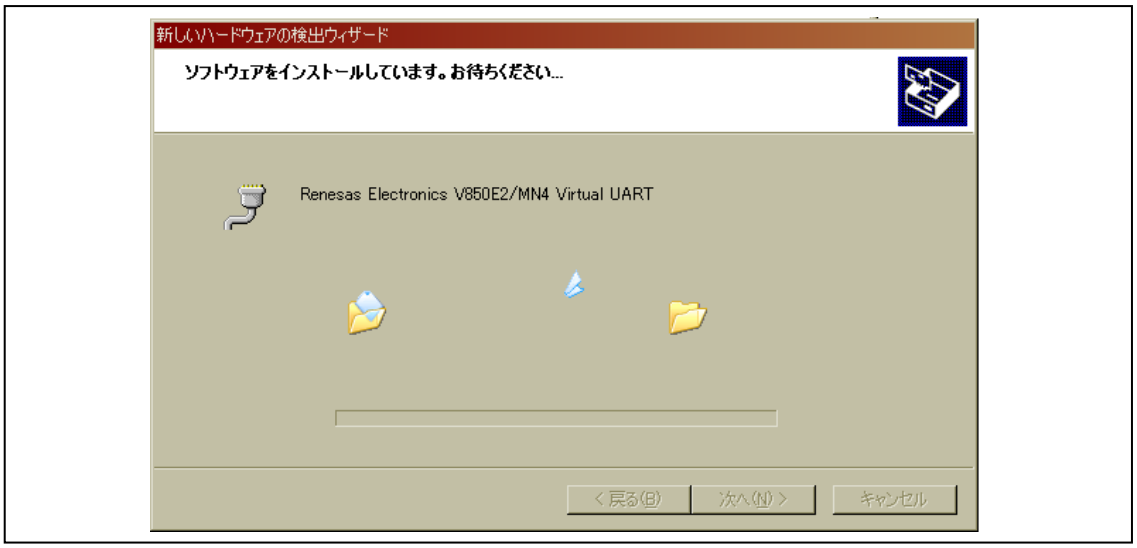


図 6－43　ドライバインストール中(1)

<11> 「ハードウエアのインストール」ダイアログが表示されます。「続行」を押下します。

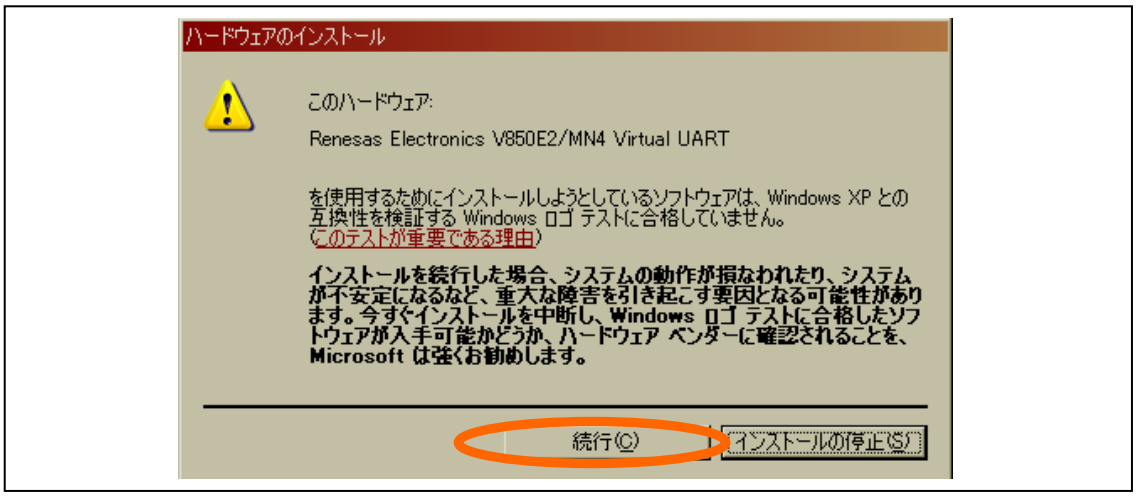


図 6－44　ドライバ互換性確認ダイアログ

<12> ドライバがインストールされます。環境により，時間がかかる場合があります。

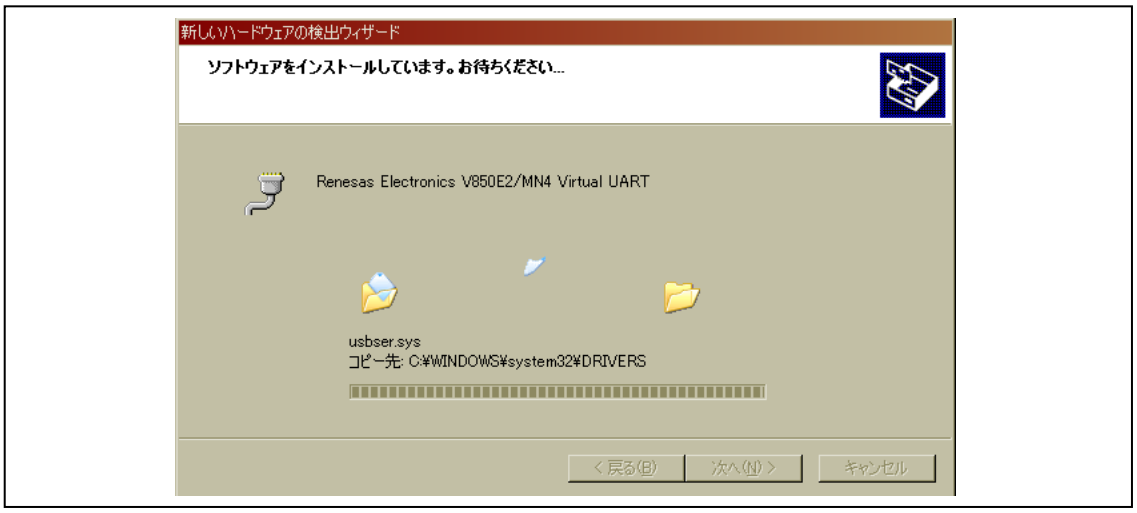


図 6－45　ドライバインストール中(2)

<13> 次の画面が表示されます。「完了」ボタンを押下します。

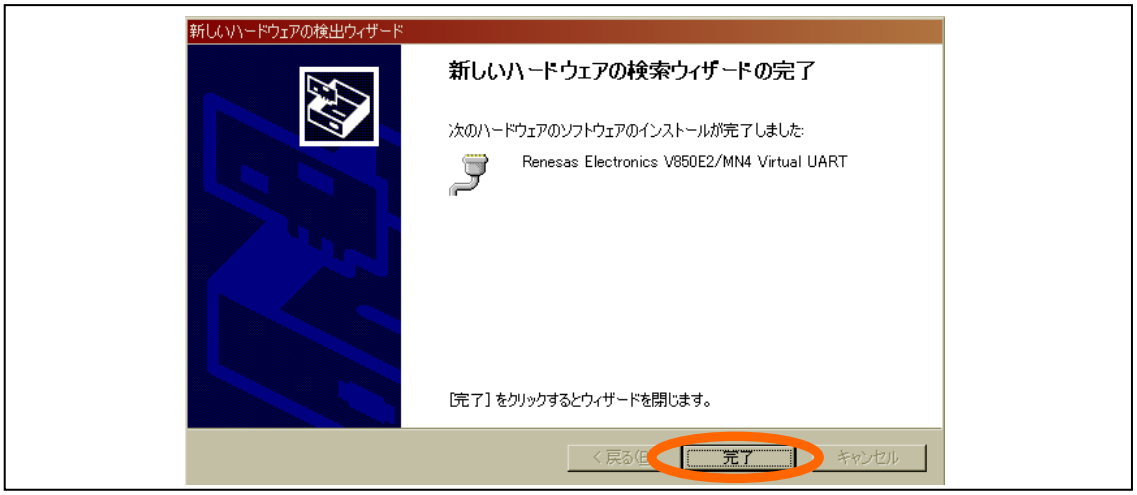


図 6－46　ドライバインストールの完了

(3) デバイス割り当ての確認

Windows のデバイスマネージャを開きます。デバイスの一覧表示の「ポート」のツリーを展開し，「Renesas Electronics V850E2/MN4 Virtual UART」が表示されていること，また割り当てられた COM ポート番号を確認します。

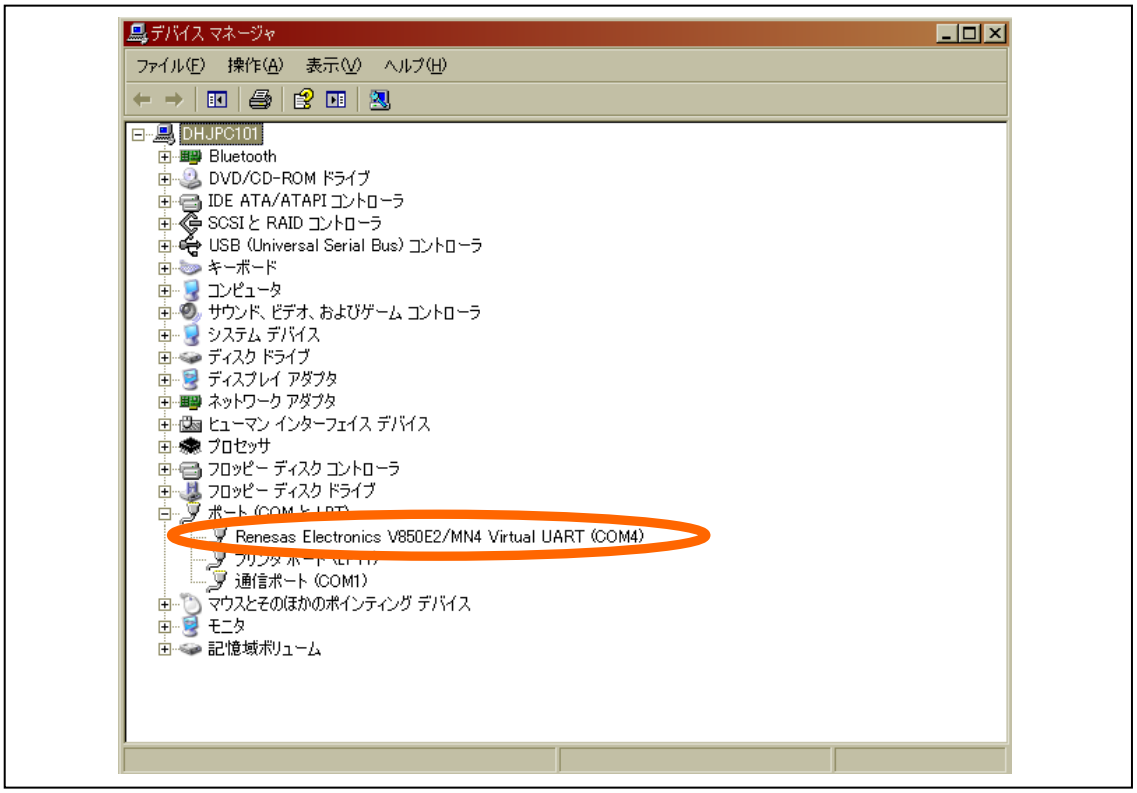


図 6－47　COM ポート確認

備考　デバイス名やポート番号は任意のものに変更できます。詳細は 7.2 カスタマイズを参照してください。

6.5　Multi環境デバッグ

ここでは，「6. 4　Multi 環境設定」に示したワークスペースで開発したアプリケーション・プログラムのデバッグ手順について説明します。

6.5.1　ロード・モジュール生成

ターゲット・デバイスにプログラムを書き込むには，C 言語やアセンブリ言語で記述されたファイルを C コンパイラなどで変換してロード・モジュールを生成します。
Multi では，「Build」メニューから「Build Top Project V850E2_MN4(CDC)_GHS.gpj」を選択すると，ロード・モジュールが生成されます。

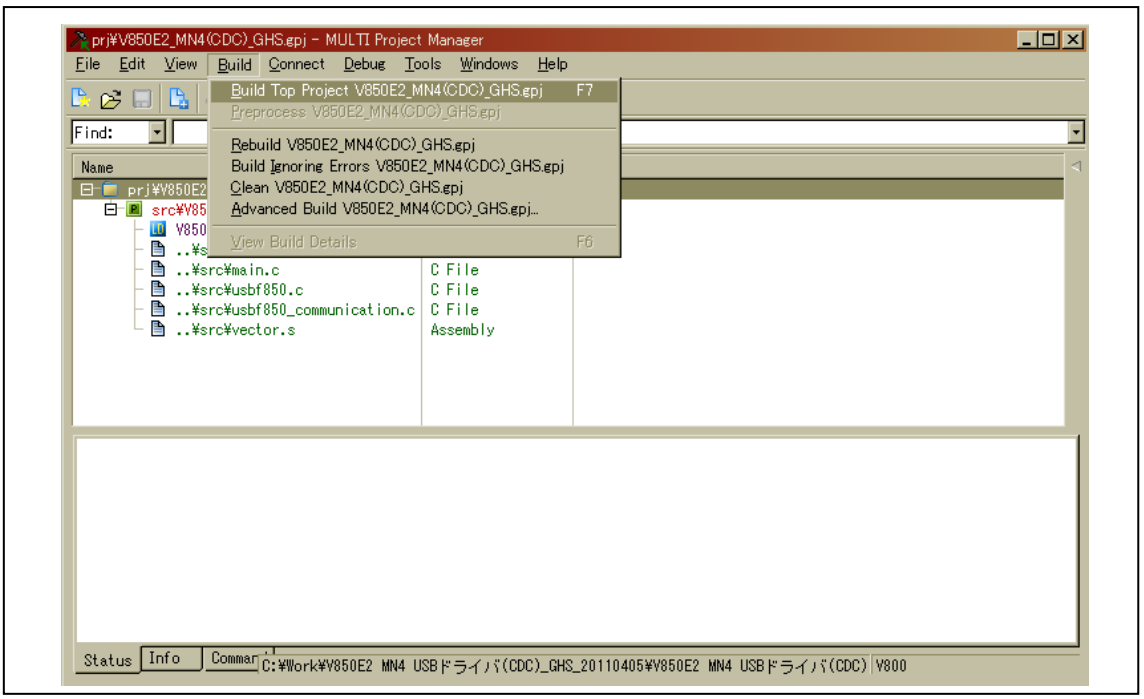


図 6－48　Build の選択

6.5.2　ロードと実行

生成したロード・モジュールをターゲットに書き込んで（ロード）実行させます。

(1) ロード・モジュールの書き込み

ここでは Multi を介して RTE-V850E2/MN4-EB-S にロード･モジュールを書き込む手順を示します。

<1>　Multi の「Connect」から「Connect」を選択し，Connection Chooser を表示します。

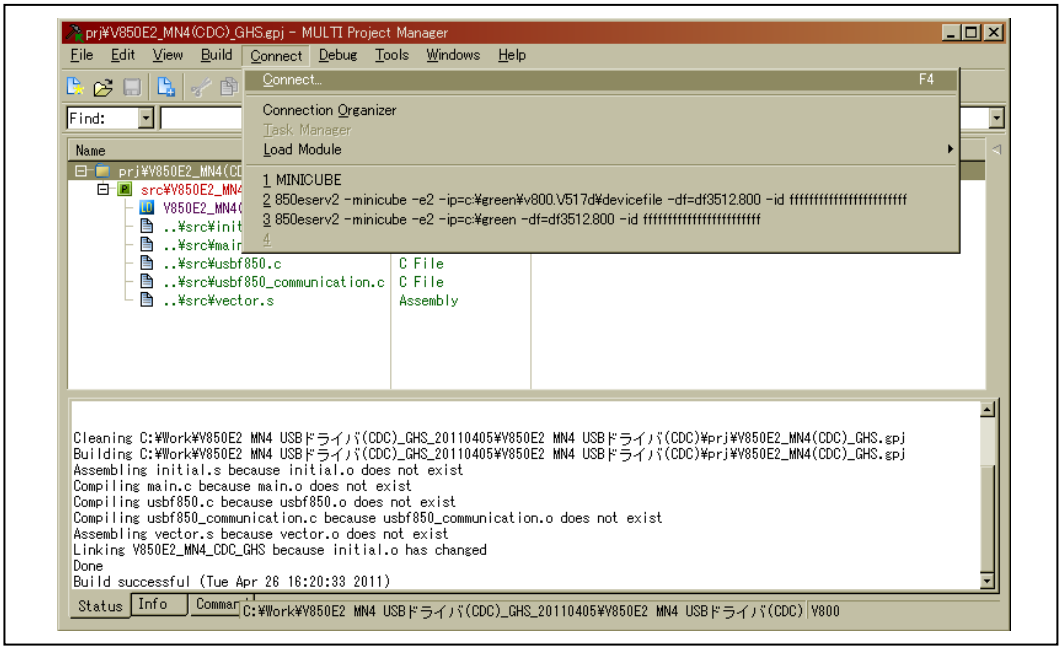


図 6－49　Connection Chooser の起動

<2>　Connection Chooser より，「6. 4. 1 ホスト環境整備」にて作成した MINICUBE 接続設定を選択して「Connect」ボタンを押下します。

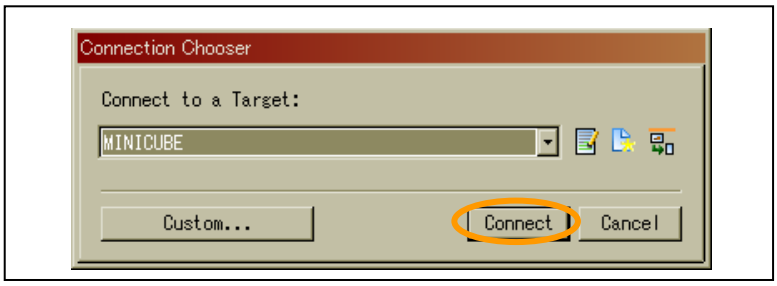


図 6－50　MINICUBE 接続設定選択

<3>　MULTI Debugger が起動するので，「File」メニューより「Debug Program」を選択し，ロード・モジュールをダウンロードします。

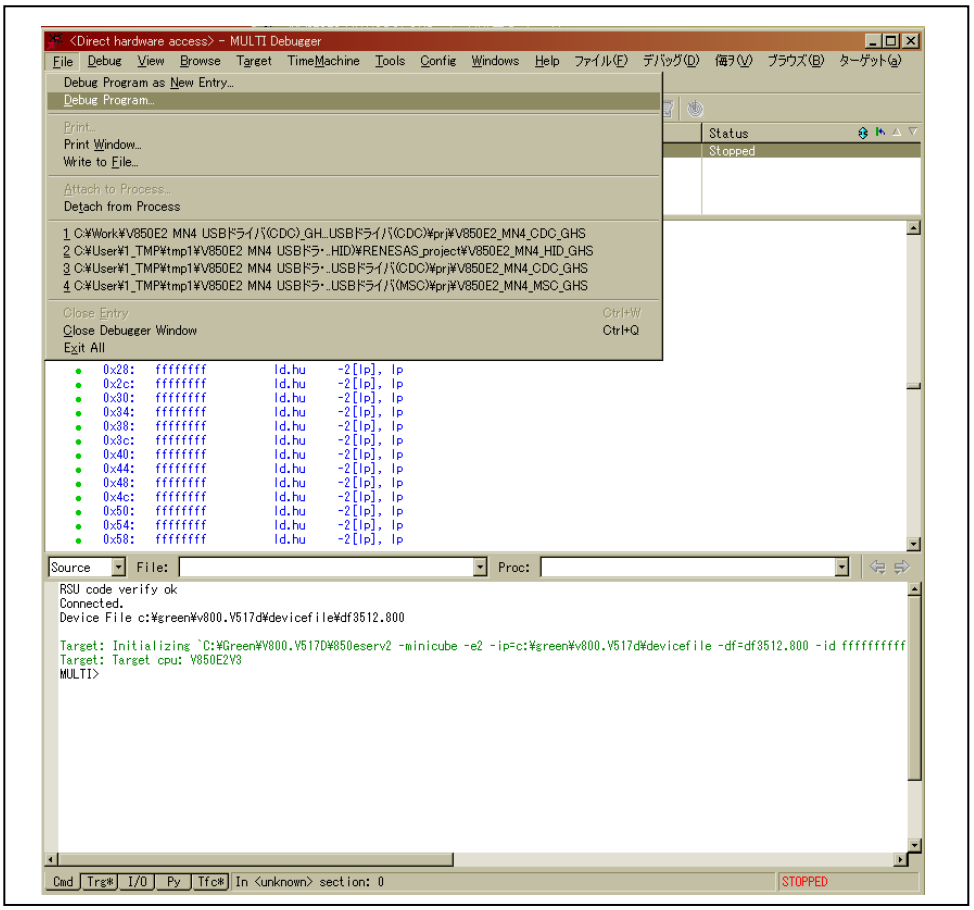


図 6－51　MULTI Debugger メニュー選択

ロード・モジュールは，「prj」フォルダに「V850E2_MN4_CDC_GHS」のファイル名で作成されているので，それを指定して「Debug」ボタンを押下します。

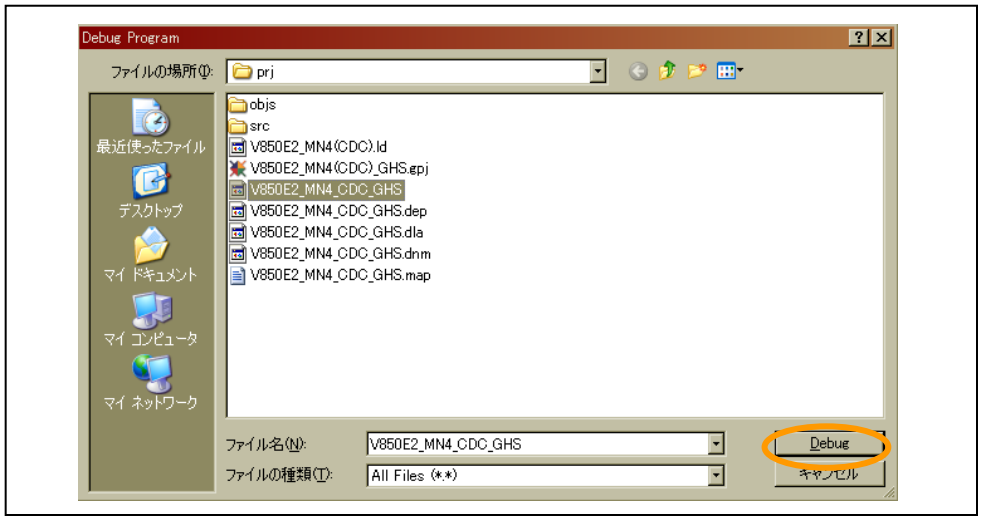


図 6－52　ロード・モジュール選択

(2) プログラムの実行

MULTI Debugger の  ボタンを押下します。または「Debug」メニューから「Go on Selected Items」を選択します。

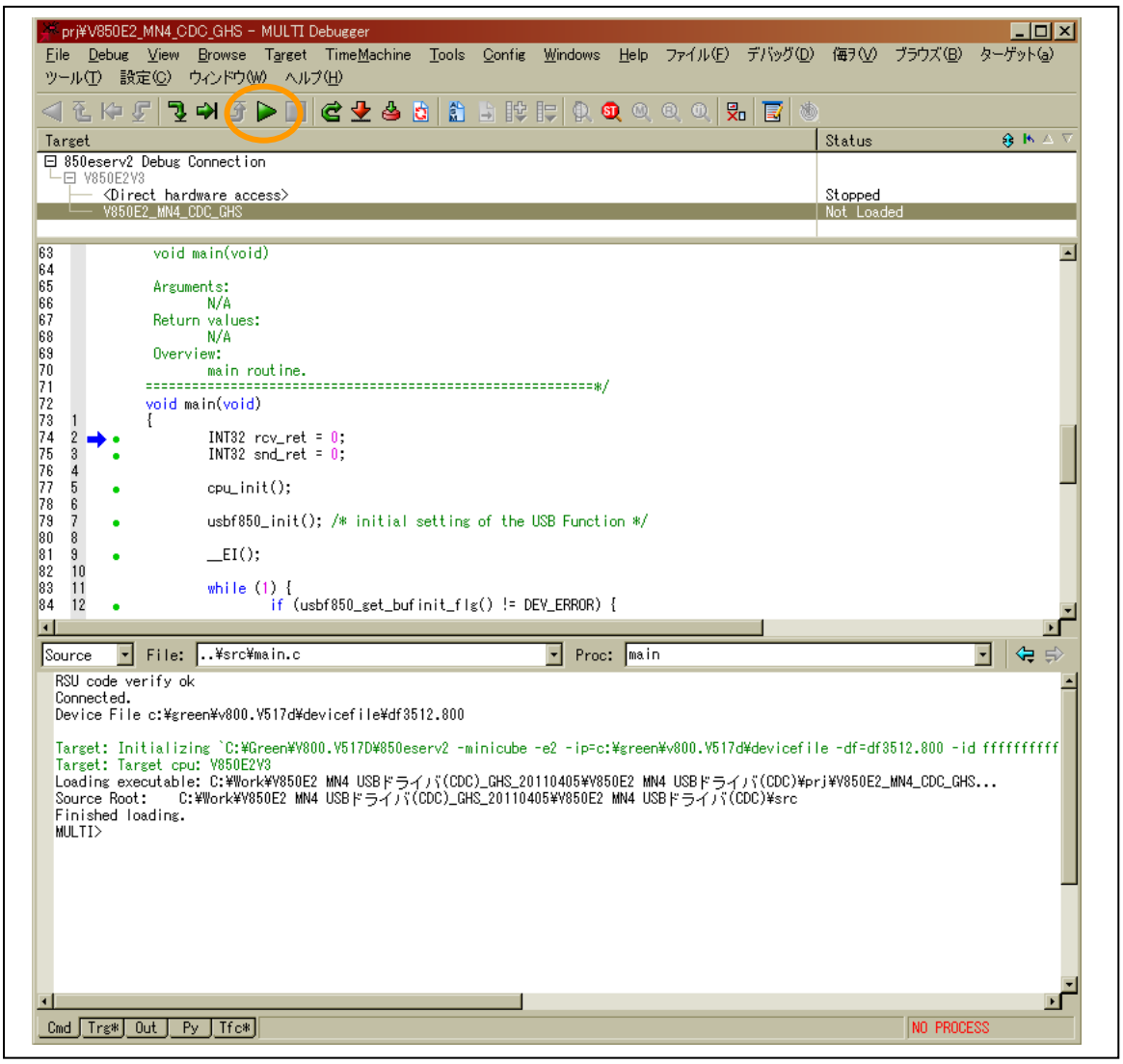


図 6－53　実行開始

6.6 IAR Embedded Workbench環境設定

ここでは，「6. 1　開発環境」に示した製品構成の中で，IAR Embedded Workbench を用いた開発やデバッグを行うための準備について説明します。

6.6.1 ホスト環境整備

ホスト・マシン上に専用のワークスペースを作成します。

(1) IAR Embedded Workbench 統合開発ツールのインストール

IAR Embedded Workbench をインストールします。詳細は IAR Embedded Workbench のユーザーズマニュアルを参照してください。

(2) ドライバ類の展開

サンプル・ドライバの提供ファイル一式を，フォルダ構成を変えずに任意のディレクトリに格納します。
また，デバッグ・ポート用ホスト・ドライバを任意のディレクトリに格納します。

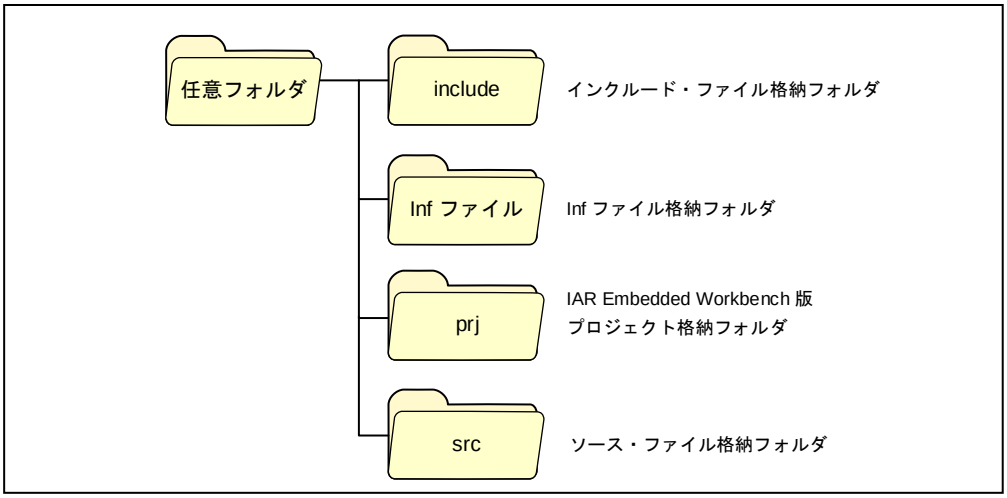


図 6－54　サンプル・ドライバのフォルダ構成（IAR Embedded Workbench 版）

(3) デバイス・ファイルのインストール

IAR Embedded Workbench 用 V850E2/MN4 用のデバイス・ファイルを，IAR Embedded Workbench インストールフォルダにコピーします。

例）　C:¥Program Files¥IAR Systems¥Embedded Workbench 5.4_0¥v850¥inc

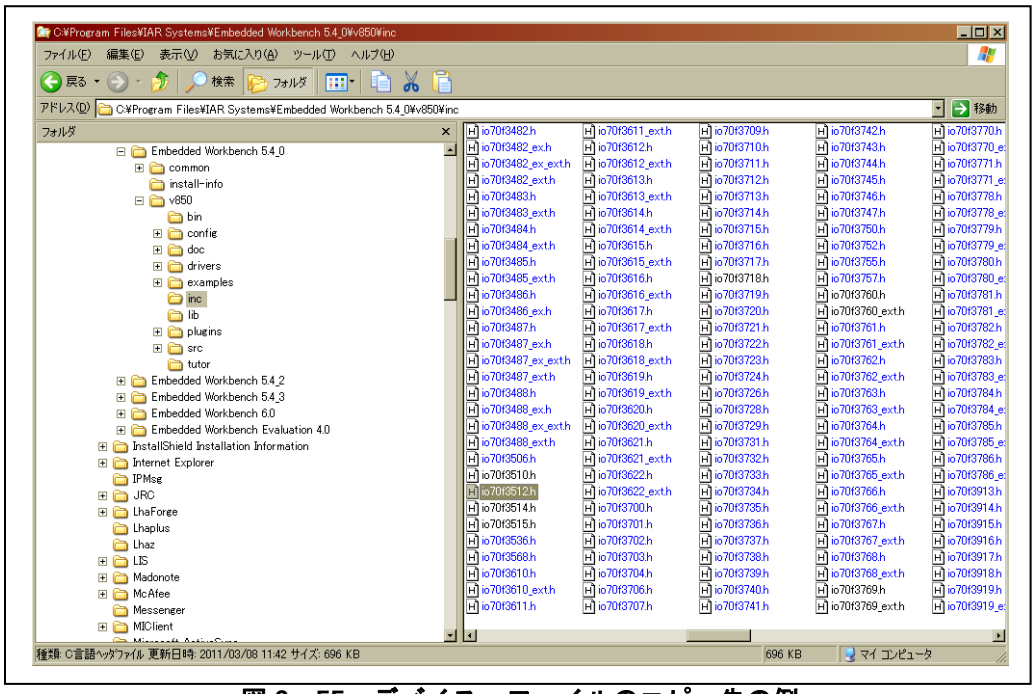


図 6－55　デバイス・ファイルのコピー先の例

(4) IAR Embedded Workbench の起動

エクスプローラから，サンプル・ドライバに同梱されている「V850E2_MN4(CDC)_IAR.eww」の IAR IDE Workspace を選択し，IAR Embedded Workbench を起動します。

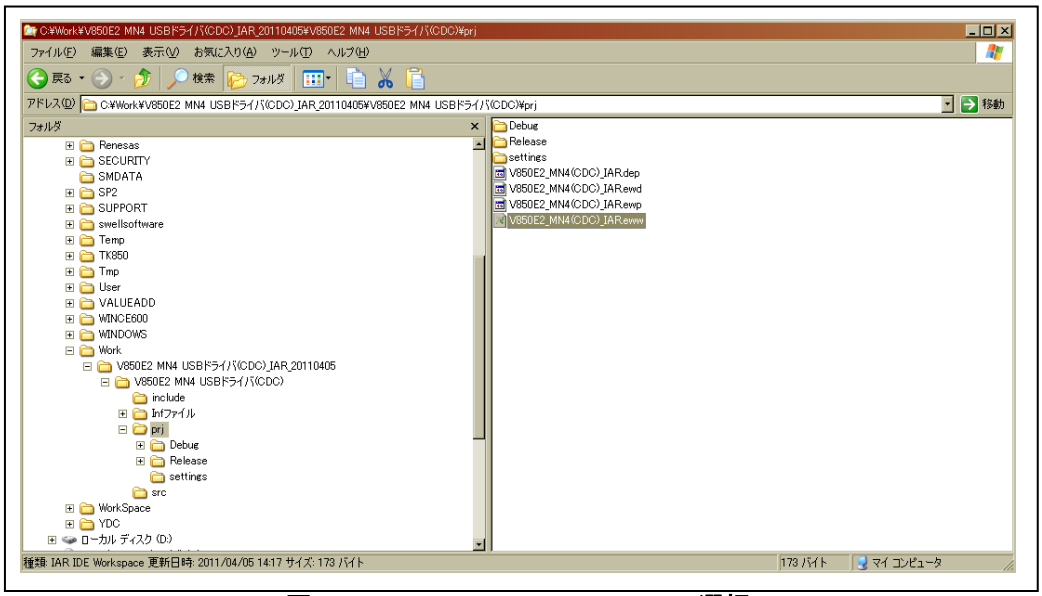


図 6－56　IAR IDE Workspace の選択

(5) デバッグ・ツールの設定

ここでは，デバッグ・ツールとして MINICUBE を使用する場合の手順を示します。

<1> IAR Embedded Workbench のプロジェクトツリーで「V850E2/MN4(CDC)_IAR-Release（または Debug）」にフォーカスを当てて右クリックを押下し「Option」を選択し，Option を表示します。

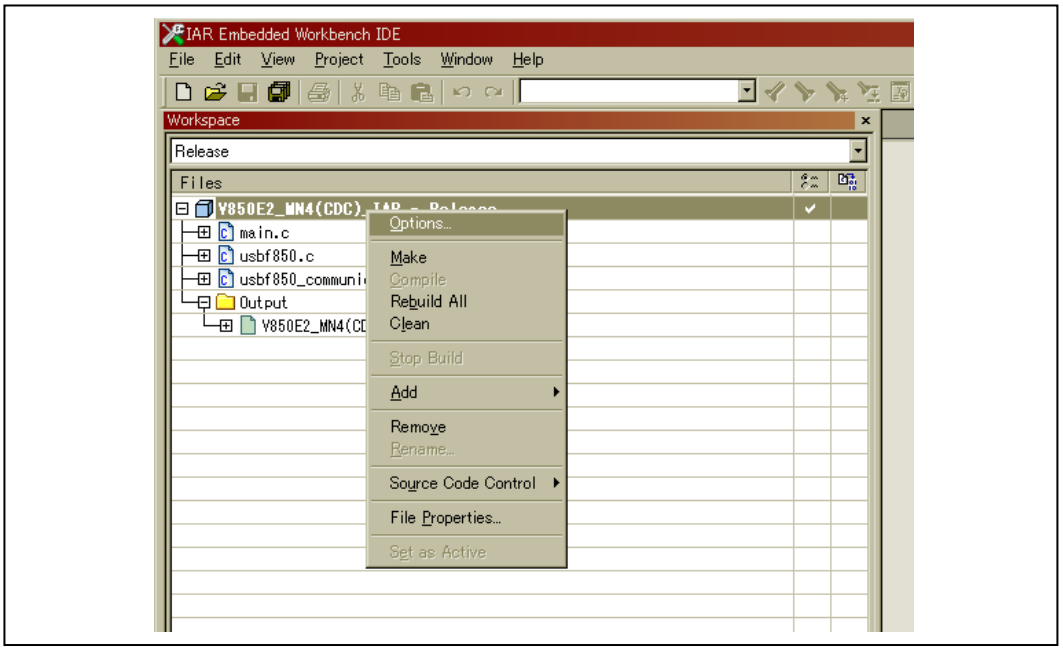


図 6－57　Option の選択

<2> Option for node “V850E2_MN4(CDC)_IAR”ダイアログから，「Category」で「Debugger」を選択します。

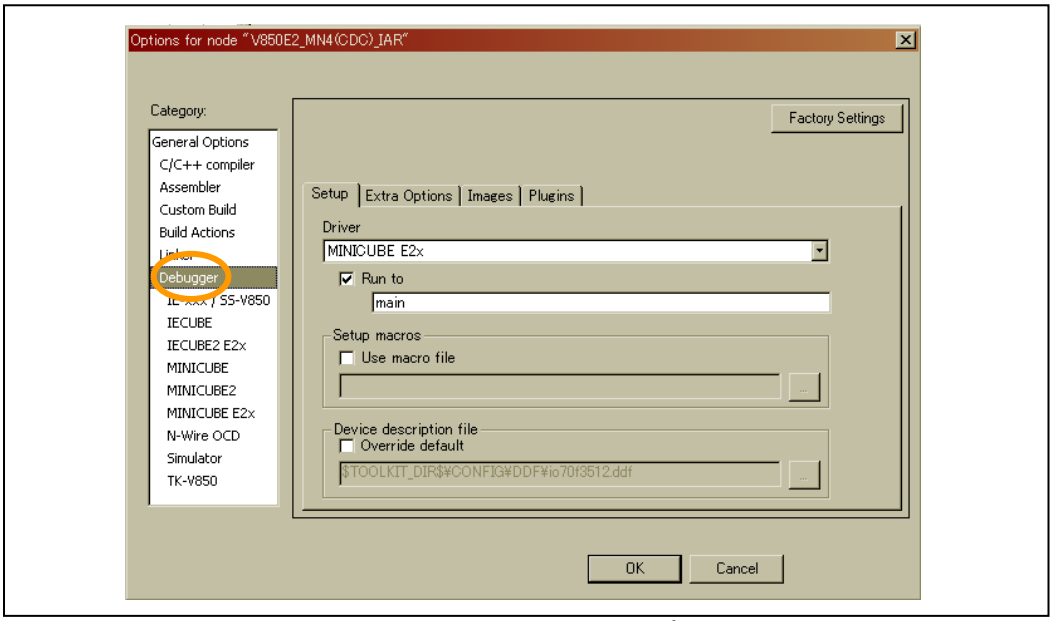


図 6－58　Debugger の選択

<3>　「Setup」タブの Driver 欄で「MINICUBE E2x」を選択して「OK」を押下します 。

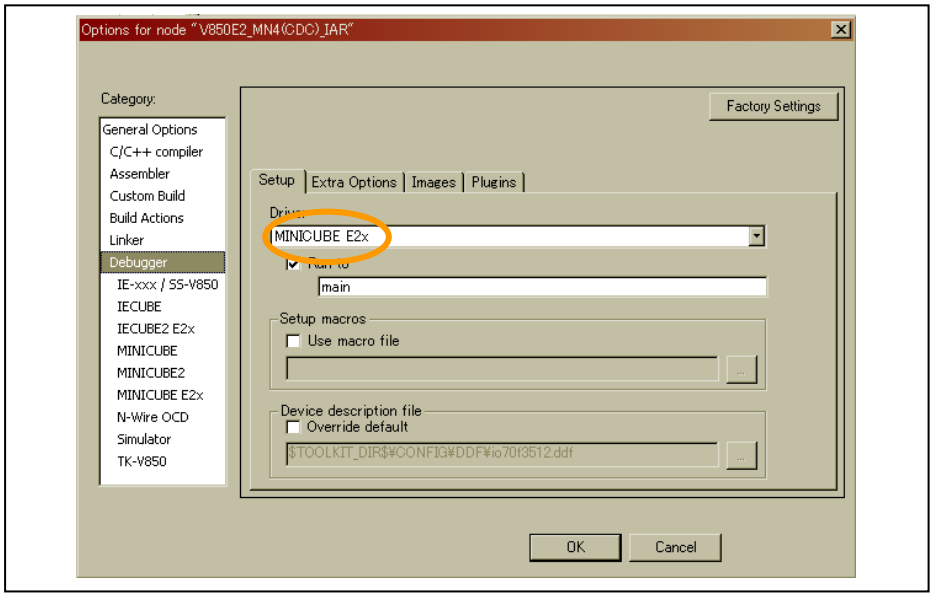


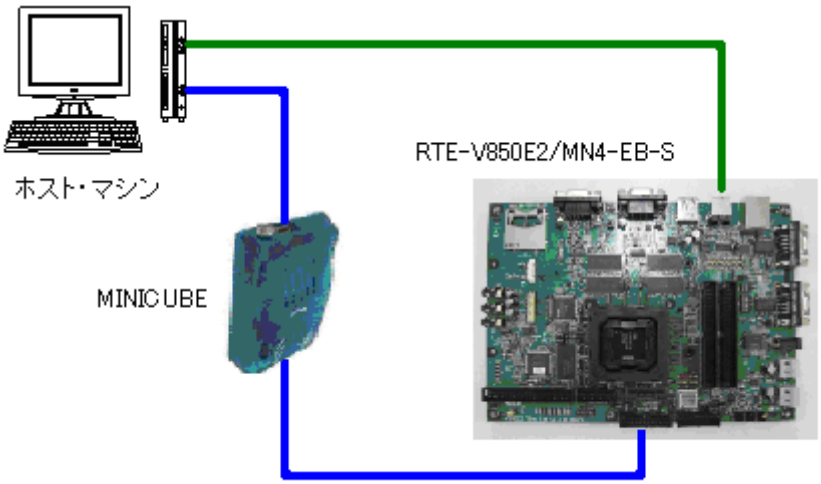
図 6－59　Debugger の選択

6.6.2　ターゲット環境整備

デバッグに使用するターゲット・デバイスを接続します。接続方法は CubeSuite/Multi/IAR Embedded Workbench 共に同じです。

(1) デバッグ・ポートの接続

RTE-V850E2/MN4-EB-S とホスト・マシンを接続します。RTE-V850E2/MN4-EB-S とホスト・マシンをデバッグ用に MINICUBE で接続します。また RTE-V850E2/MN4-EB-S の USB B タイプコネクタとホスト・マシンの USB コネクタを CDC 用に接続します。



（備考）RTE-V850E2/MN4-EB-S の外形やポートについては，「8. スタータ・キット概説」を参照してください。

図 6－60　RTE-V850E2/MN4-EB-S の接続

(2) ホスト・ドライバのインストール

ここではサンプル・ドライバに同梱の仮想 COM ポート用ホスト・ドライバを使用する場合の手順を示します。

<1> RTE-V850E2/MN4-EB-S の接続がホスト・マシンに認識されると，「新しいハードウエアが見つかりました」というメッセージが表示されたあと，新しいハードウエアの検出ウィザードが起動します。

<2> 「新しいハードウエアの検出ウィザード」ダイアログが開きます。「いいえ、今回は接続しません」を選択して「次へ」ボタンを押下します。

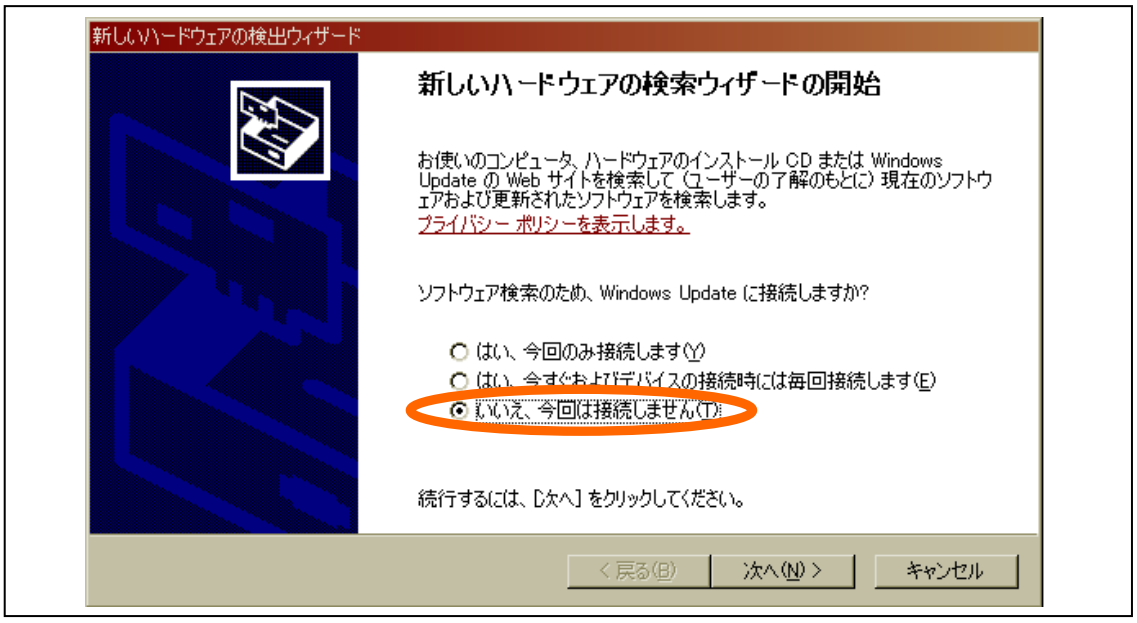


図 6－61　新しいハードウエアの検出ウィザード(1)

<3> 次の画面が表示されます。「一覧または特定の場所からインストールする（詳細）」を選択して「次へ」ボタンを押下します。

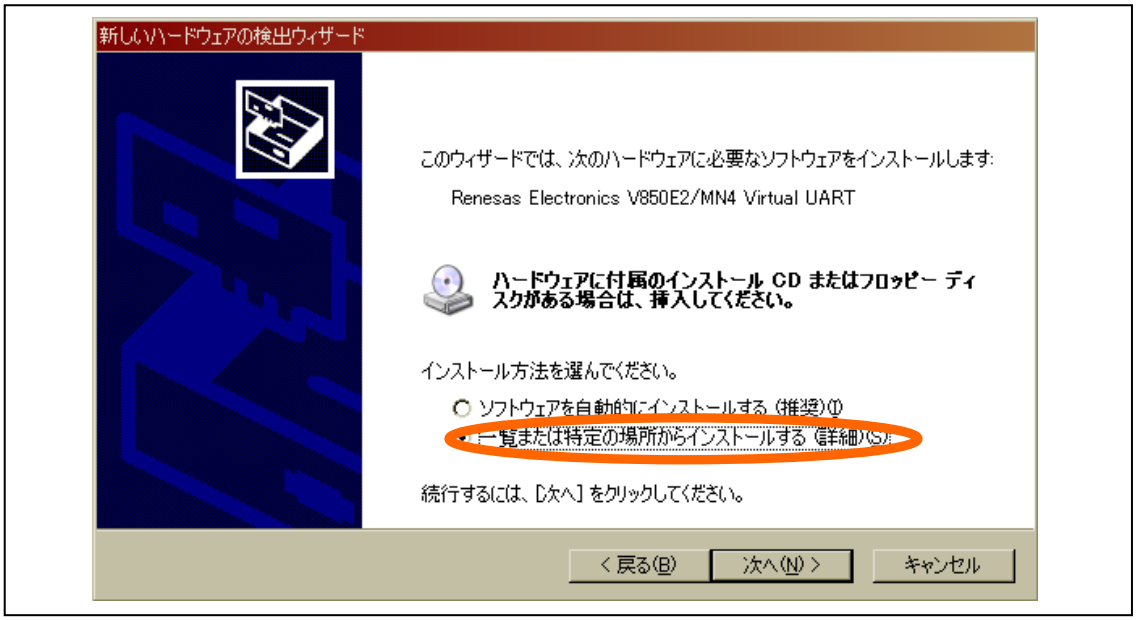


図 6－62　新しいハードウエアの検出ウィザード(2)

<4>　次の画面が表示されます。「検索しないで、インストールするドライバを選択する」を選択して「次へ」ボタンを押下します。

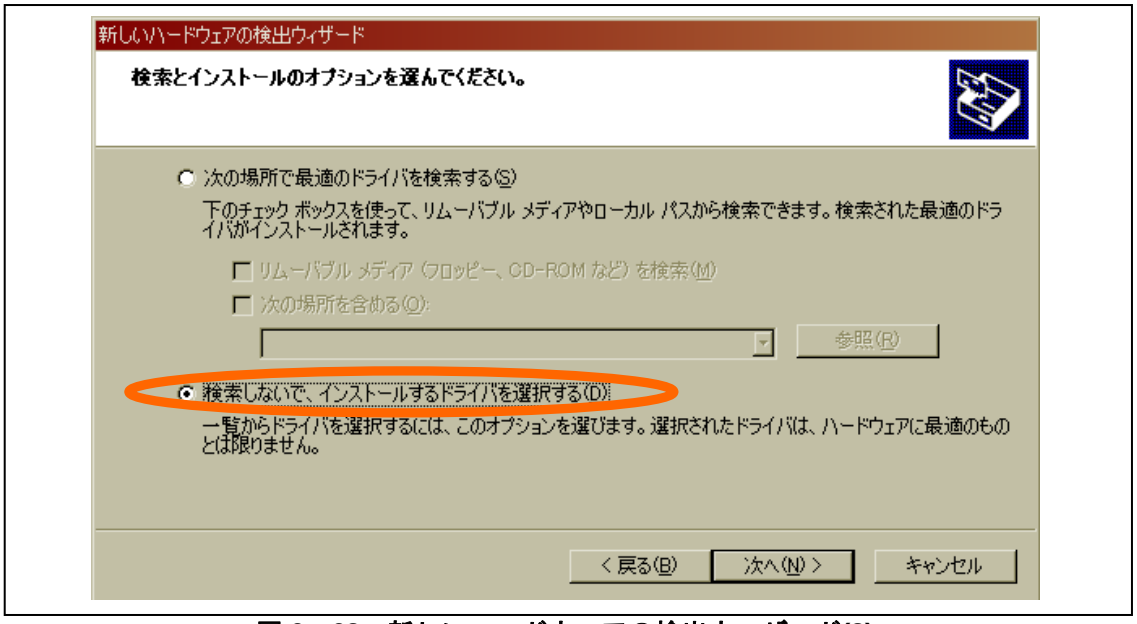


図 6－63　新しいハードウエアの検出ウィザード(3)

<5>　次の画面が表示されます。「ディスク使用」ボタンを押下します。

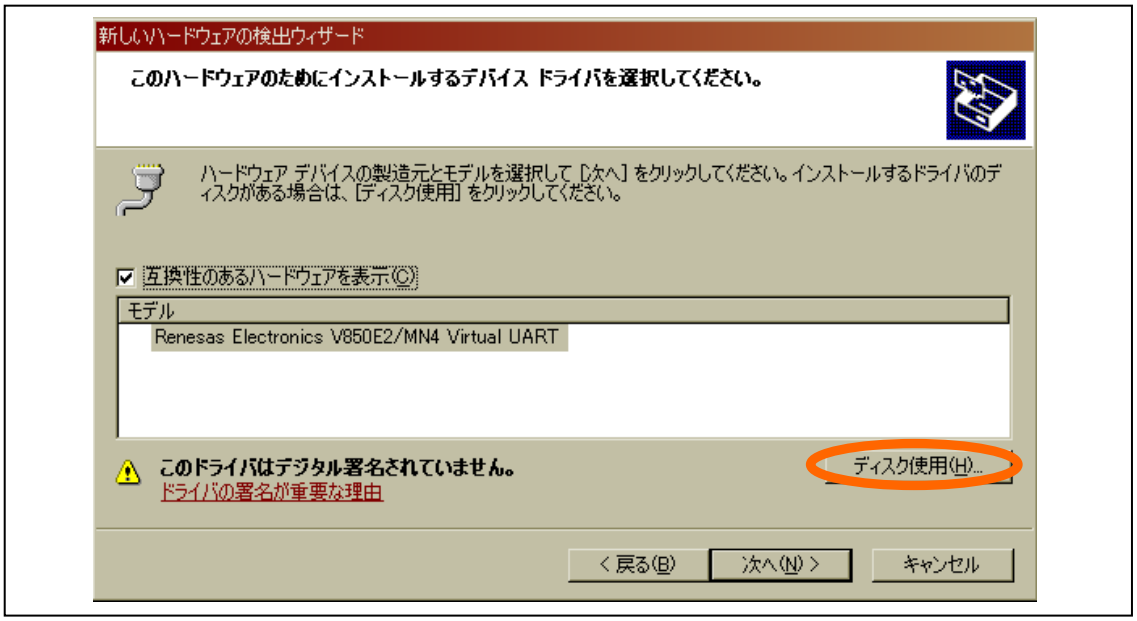


図 6－64　新しいハードウエアの検出ウィザード(4)

<6>　「フロッピーディスクからインストール」ダイアログが開きます。「参照」ボタンを押下して，サンプル・ドライバを格納したディレクトリの「Inf ファイル」フォルダを表示させます。

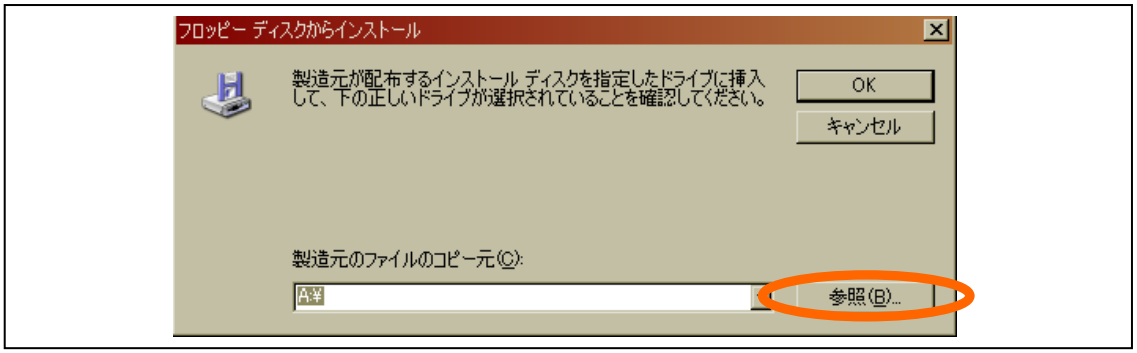


図 6－65　Inf ファイルの参照

<7>　ホスト・マシンで使用している OS にあわせ「XP」「VISTA」「Win7」フォルダの中の INF ファイルを選択し，「開く」ボタンを押下します。

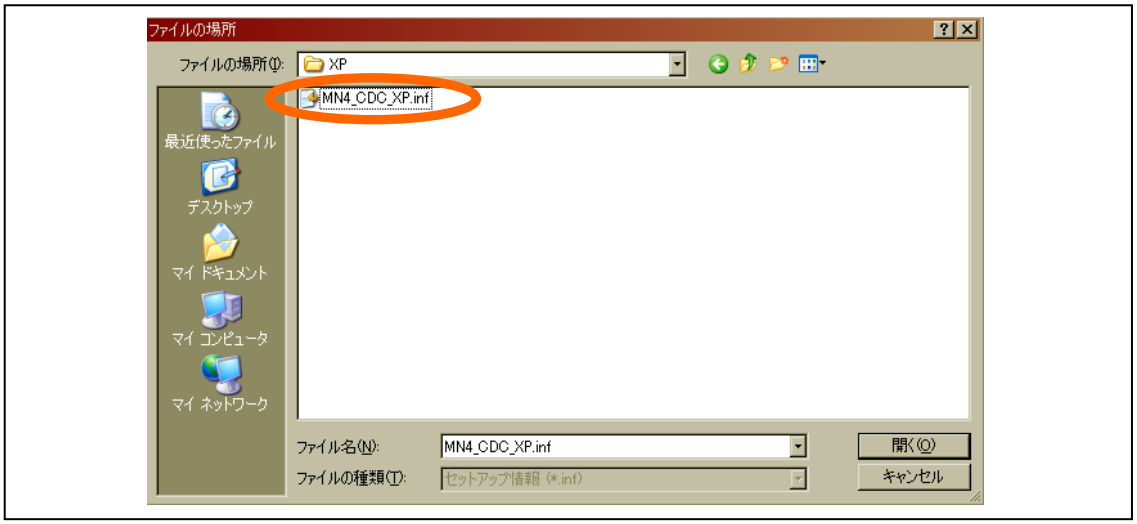


図 6－66　Inf ファイル選択

<8>　「フロッピーディスクからインストール」ダイアログに戻ります。「製造元のファイルのコピー元」欄が正しいことを確認し，「OK」ボタンを押下します。

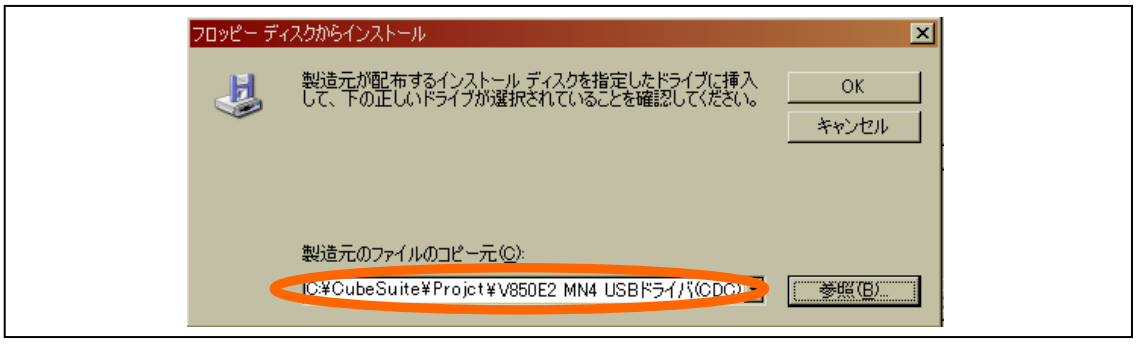


図 6－67　Inf ファイルインストール

<9>　「新しいハードウエアの検出ウィザード」画面に戻ります。「モデル」欄で「Renesas Electronics V850E2/MN4 Virtual UART」を選択して「次へ」ボタンを押下します。

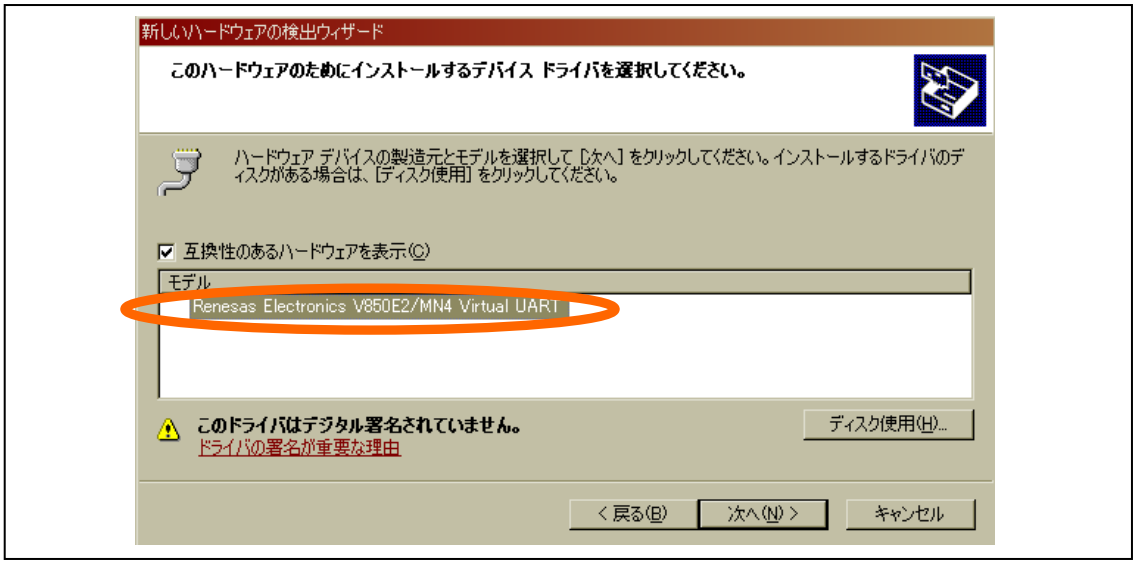


図 6－68　インストールデバイスドライバ選択

<10>　ドライバのインストールが開始されます。

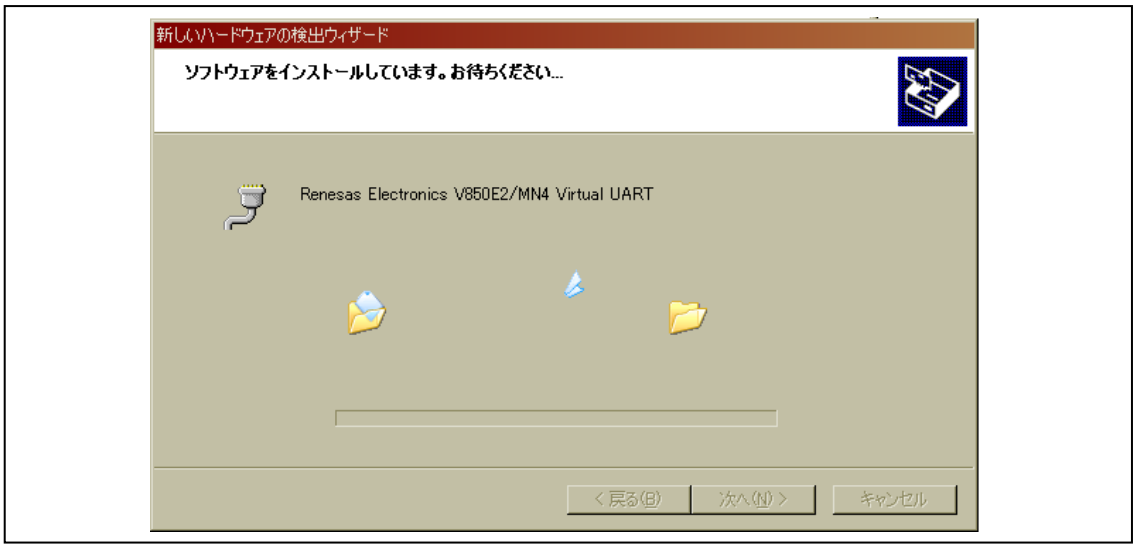


図 6－69　　ドライバインストール中(1)

<11> 「ハードウエアのインストール」ダイアログが表示されます。「続行」を押下します。

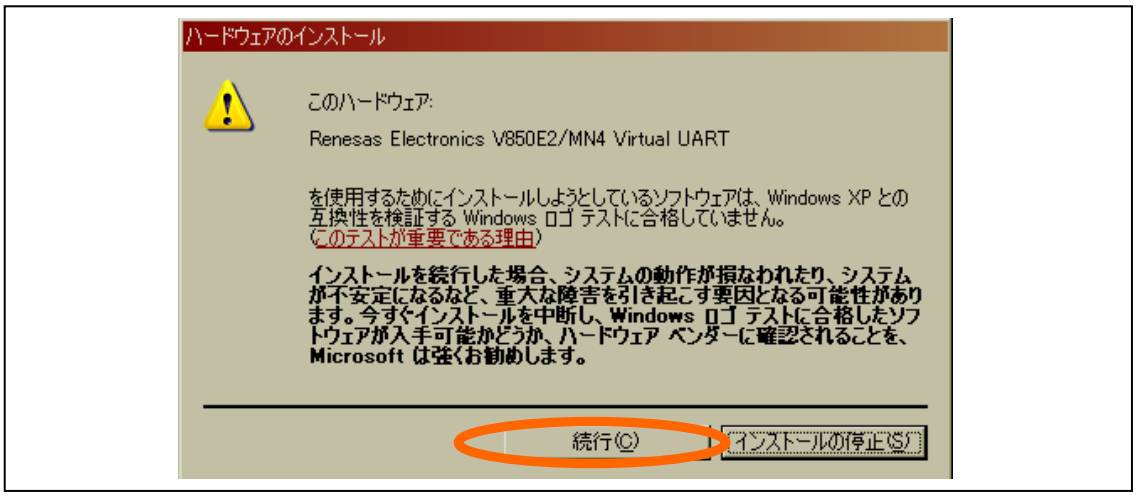


図 6－70　ドライバ互換性確認ダイアログ

<12> ドライバがインストールされます。環境により，時間がかかる場合があります。

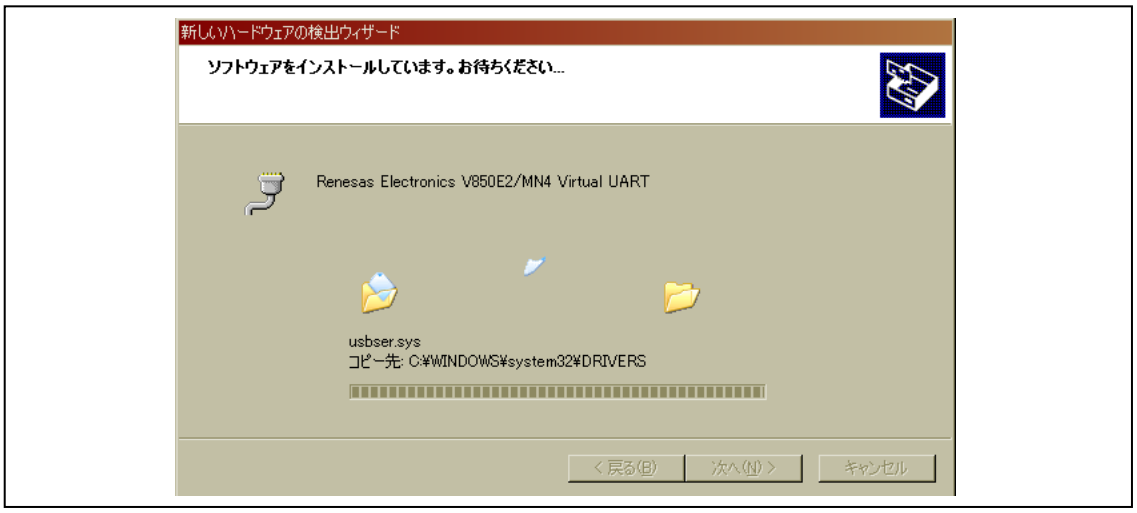


図 6－71　ドライバインストール中(2)

<13> 次の画面が表示されます。「完了」ボタンを押下します。

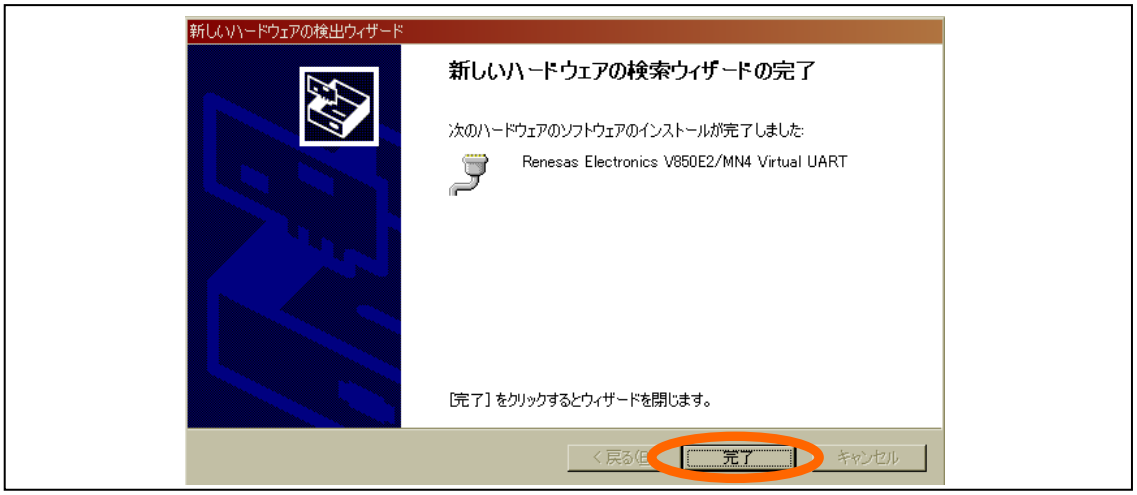


図 6－72　ドライバインストールの完了

(3) デバイス割り当ての確認

Windows のデバイスマネージャを開きます。デバイスの一覧表示の「ポート」のツリーを展開し，「Renesas Electronics V850E2/MN4 Virtual UART」が表示されていること，また割り当てられた COM ポート番号を確認します。

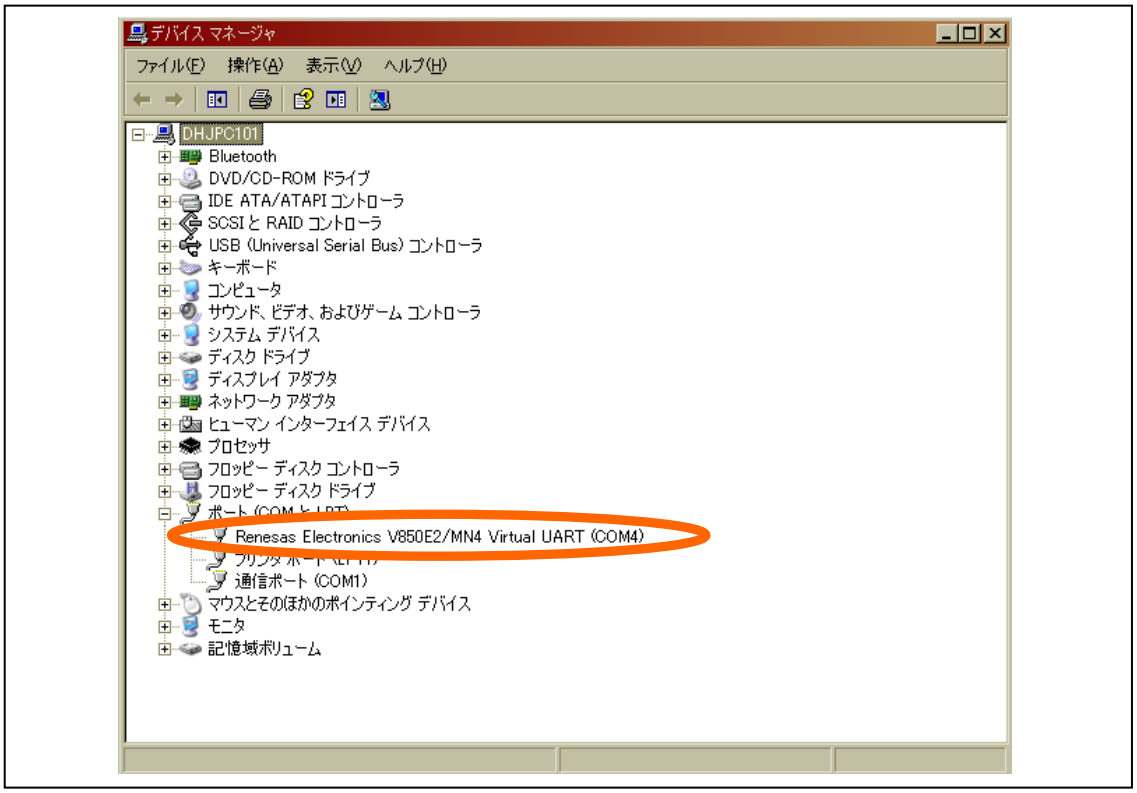


図 6－73　COM ポート確認

備考　デバイス名やポート番号は任意のものに変更できます。詳細は 7.2 カスタマイズを参照してください。

6.7 IAR Embedded Workbench環境デバッグ

ここでは,「6. 6　IAR Embedded Workbench 環境設定」に示したワークスペースで開発したアプリケーション・プログラムのデバッグ手順について説明します。

6.7.1 ロード・モジュール生成

ターゲット・デバイスにプログラムを書き込むには，C 言語やアセンブリ言語で記述されたファイルを C コンパイラなどで変換してロード・モジュールを生成します。
IAR Embedded Workbench では，「Project」メニューから「Rebuild all」を選択すると，ロード・モジュールが生成されます。

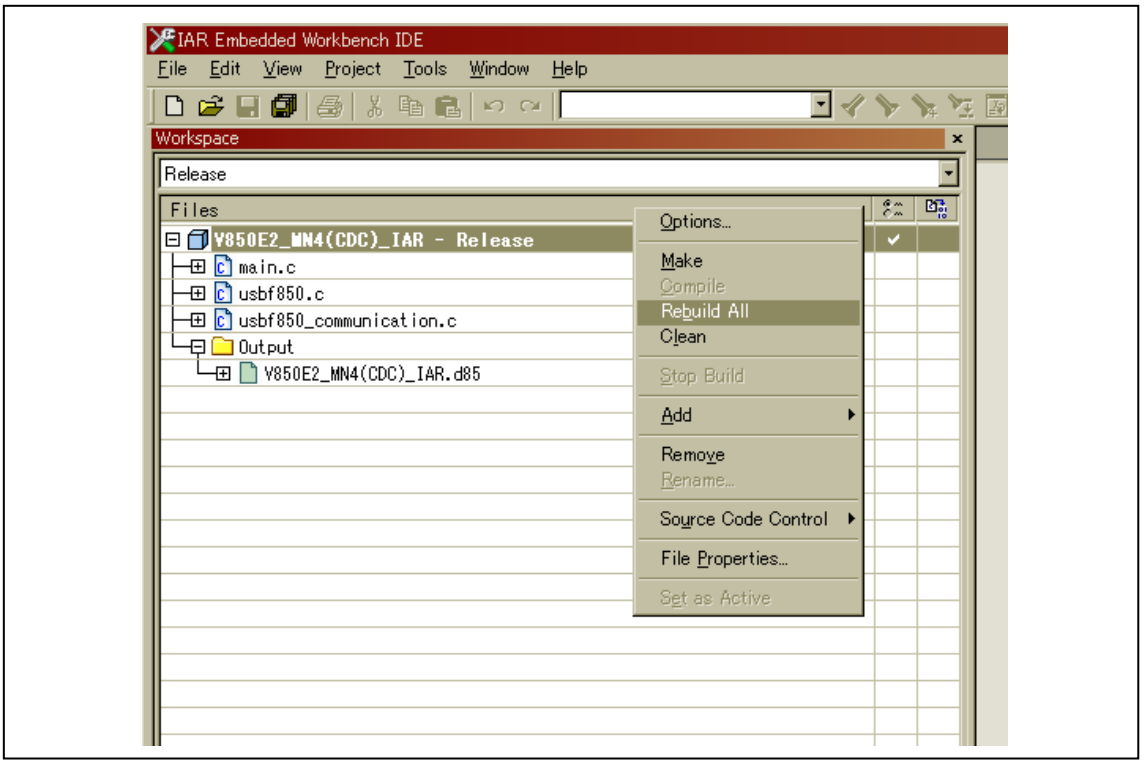


図 6－74　Rebuild all の選択

6.7.2　ロードと実行

生成したロード・モジュールをターゲットに書き込んで（ロード）実行させます。

(1) ロード・モジュールの書き込み

ここでは IAR Embedded Workbench を介して RTE-V850E2/MN4-EB-S にロード・モジュールを書き込む手順を示します。

<1>　IAR Embedded Workbench の「Project」から「Download and Debug」を選択し，ターゲットにロードします。

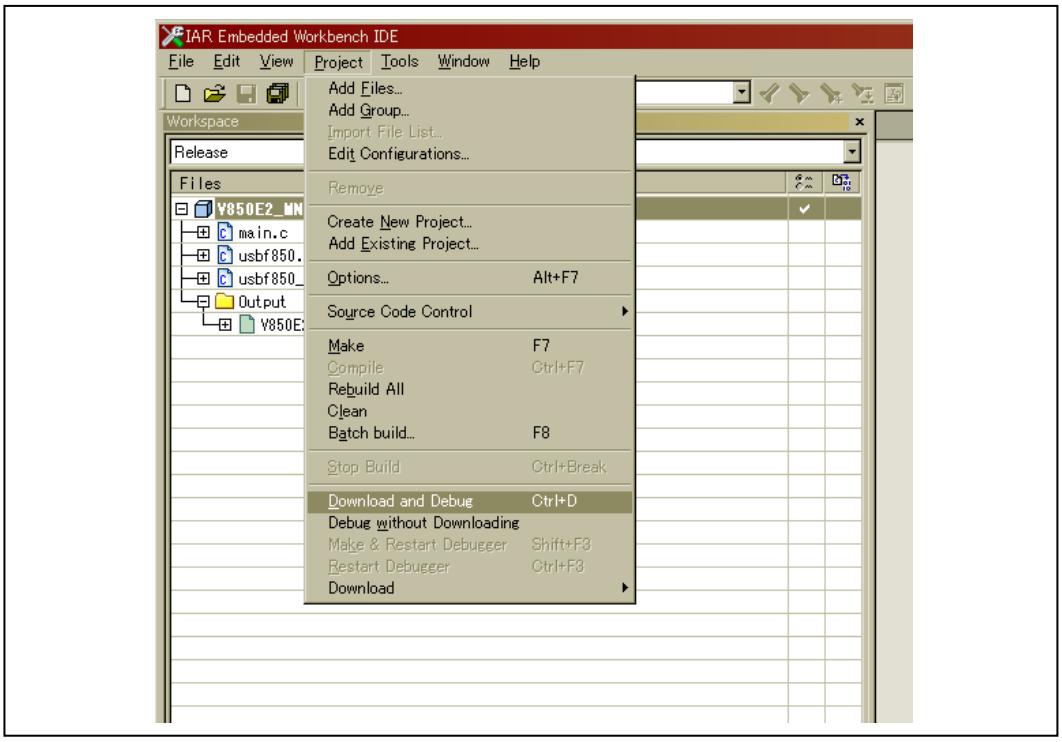


図 6－75　Debugger の起動

<2>　デバッグ・ツールを介して，ロード・モジュールのダウンロードが開始されます。

(2) プログラムの実行

IAR Embedded Workbench の  ボタンを押下します。または「Debug」メニューから「Go」を選択します。

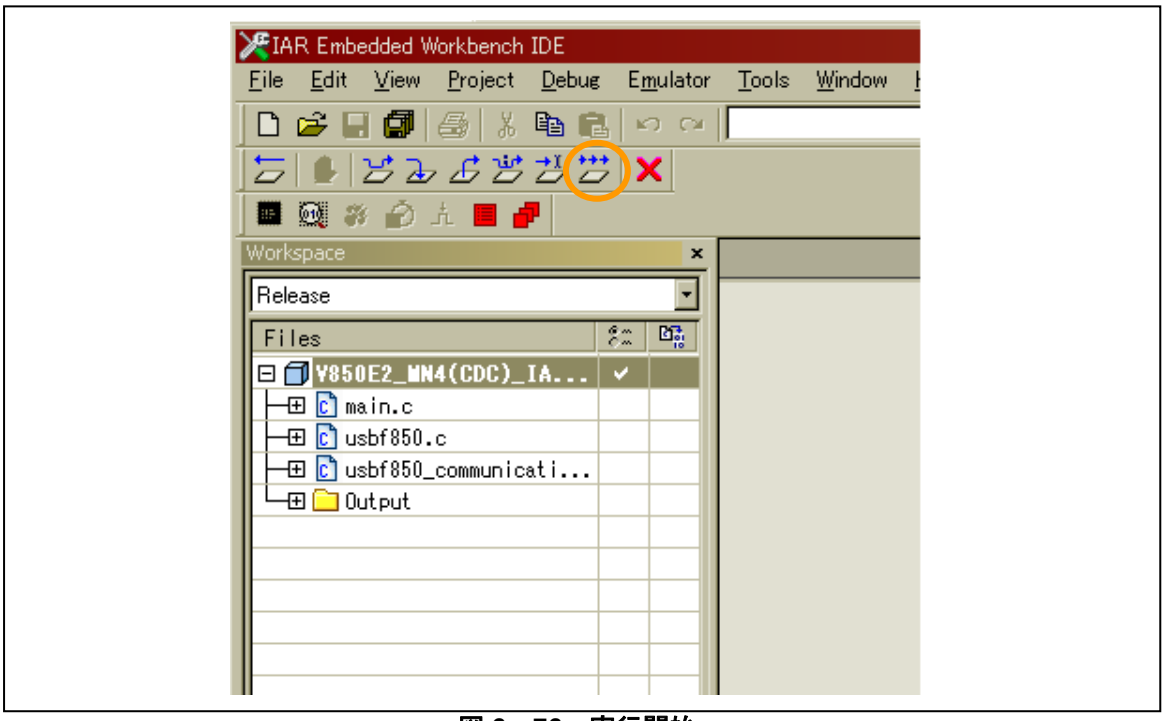


図 6－76　実行開始

6.8 動作確認

サンプル・ドライバをロードしたターゲット・デバイスとホスト・マシンが USB で接続されていれば，ドライバ内のサンプル・アプリケーションの実行結果を確認できます。
ホスト上でターミナル・ソフトウエア（Tera Term など）を起動し，入力したものがそのまま表示される事を確認して下さい。

備考　サンプル・アプリケーションの詳細は第 5 章 サンプル・アプリケーションの仕様を参照してください。

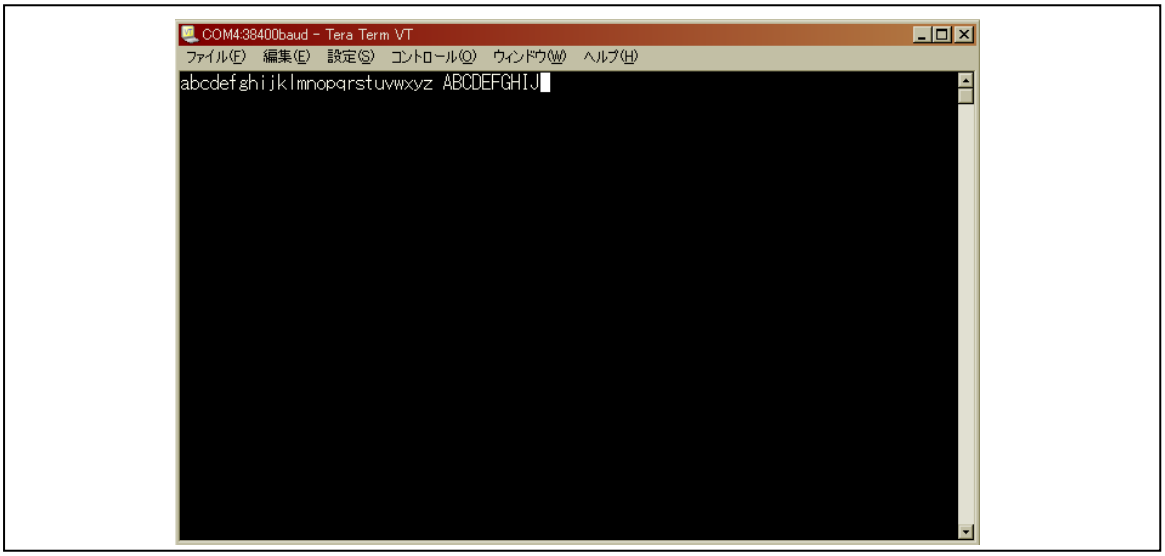


図 6－77　ターミナル入力文字のエコー結果

7. サンプル・ドライバの応用

この章では，V850E2/MN4 マイクロコンピュータ向け USB コミュニケーション・デバイス・クラス用サンプル・ドライバを利用する際に，知っておいていただきたい情報について説明します。

7.1 概　要

サンプル・ドライバ利用の際に，通常、編集する必要が生じるのは以下の部分になります。

(1) カスタマイズ

次に示す部分を必要に応じて書き換えます。

- "main.c" ファイル内のサンプル・アプリケーション部
- "usbf850.h" ファイル内の各種レジスタの設定値
- "usbf850_desc.h" ファイル内の各種ディスクリプタの内容
- 仮想 COM ポート用ホスト・ドライバ（INF ファイル）内のデバイス名やプロバイダ情報

備考　サンプル・ドライバのファイル構成については 2. 1. 3 サンプル・ドライバの構成を参照してください。

(2) 関数の利用

アプリケーション・プログラム内で必要に応じて呼び出します。実装されている関数の詳細は，4. 3 関数の仕様を参照してください。

7.2 カスタマイズ

ここでは，サンプル・ドライバの利用にあたり，必要に応じて書き換える部分について説明します。

7.2.1 アプリケーション部

"main.c" ファイルの次に示す部分は，サンプル・ドライバの利用例として簡単な処理を記述したものです。実際にアプリケーションとして必要な処理をこの部分に適宜，記述して下さい。その前にある初期化処理関数やアプリケーション内にあるデータ送受信関数はそのまま利用する事も出来ます。使用の際には 4.3 関数の仕様， 7.3 関数の利用を確認して使用して下さい。

```
1   /*============================================================
2    Main function
3    void main(void)
4
5    Arguments:
6       N/A
7    Return values:
8       N/A
9    Overview:
10      main routine.
11  ============================================================*/
12  void main(void)
13  {
14      INT32 rcv_ret = 0;
15      INT32 snd_ret = 0;
16
17      cpu_init();
18
19      usbf850_init();/* initial setting of the USB Function */
20
21      __EI();
22
23      while (1) {
24          if (usbf850_get_bufinit_flg() != DEV_ERROR) {
25              if (snd_ret >= 0) {
26                  rcv_ret = usbf850_recv_buf(&UserBuf[0], USERBUF_SIZE);
27              }
28              if (rcv_ret >= 0) {
29                  snd_ret = usbf850_send_buf(&UserBuf[0], rcv_ret);
30              }
31          }
32          else {
33              snd_ret = 0;
34              rcv_ret = 0;
35          }
36
37          if (usbf850_rsuspd_flg == SUSPEND) {
38
39              __DI();
40
41              __halt();
42
43              usbf850_rsuspd_flg = RESUME;
44
45              __EI();
46          }
47      }
48  }
```

図 7−1 サンプル・アプリケーションの記述

7.2.2 レジスタの設定

サンプル・ドライバが使用する（書き込みを行う）レジスタとその設定値は，"usbf850.h" ファイルに定義されています。これらのファイル内の値を実際のアプリケーションでの使用法に合わせて書き換えることで，サンプル・ドライバを通してターゲット・デバイスの動作を設定できます。

（1）"usbf850.h" ファイル

USB ファンクション・コントローラのレジスタ設定値の定義が記述されています。

7.2.3 ディスクリプタの内容

初期化処理時にサンプル・ドライバが USB ファンクション・コントローラに登録するデータ（4. 1. 3 ディスクリプタの設定参照）が "usbf850_desc.h" ファイルに定義されています。このファイル内の値を実際のアプリケーションでの使用法に合わせて書き換えることで，サンプル・ドライバを通してターゲット・デバイスの属性などの情報を設定できます。
なお，デバイス・ディスクリプタのベンダ ID やプロダクト ID を書き換えた場合は，ターゲット・デバイス接続の際にインストールするホスト・ドライバ（INF ファイル）でも同じように書き換える必要があります（7. 2. 4（3）ベンダ ID とプロダクト ID の変更参照）。
また，ストリング・ディスクリプタには任意の情報を登録できます。サンプル・ドライバでは製造元や製品情報を定義していますので，適宜，書き換えてください。

7.2.4　仮想COMポート用ホスト・ドライバの設定

6. 2. 2 ターゲット環境整備でインストールしたドライバに関連して，次のようなカスタマイズが可能です。

（1）COM ポート番号の変更

USB デバイスの接続を認識すると，そのデバイスの COM ポート番号をホストが自動的に割り付けますが，任意の番号に変更することもできます。ホスト・マシンで COM ポート番号を変更する手順は次のとおりです。

<1>　Windows のデバイスマネージャを開き，デバイスの一覧表示の「ポート」のツリーを展開します。

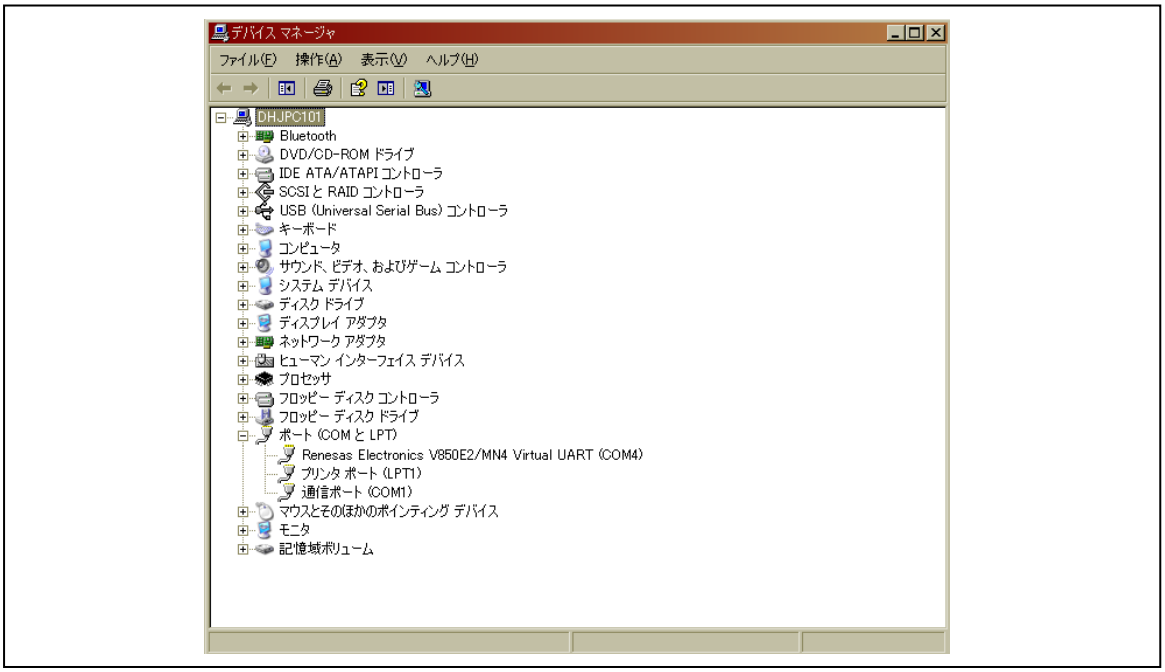


図 7−2　デバイスマネージャ表示

<2>　「Renesas Electronics V850E2/MN4 Virtual UART (COMn)」（n はホストが割り付けた番号）を選択してプロパティを表示します。

<3>　「ポートの設定」タブの「詳細設定」ボタンを押下します。

図 7−3　ポートの設定タブ表示

<4>　「COMn の詳細設定」ダイアログ（n はホストが割り付けた番号）が開きます。「COM ポート番号」欄のドロップダウン・リストから任意のポート番号を選択します。

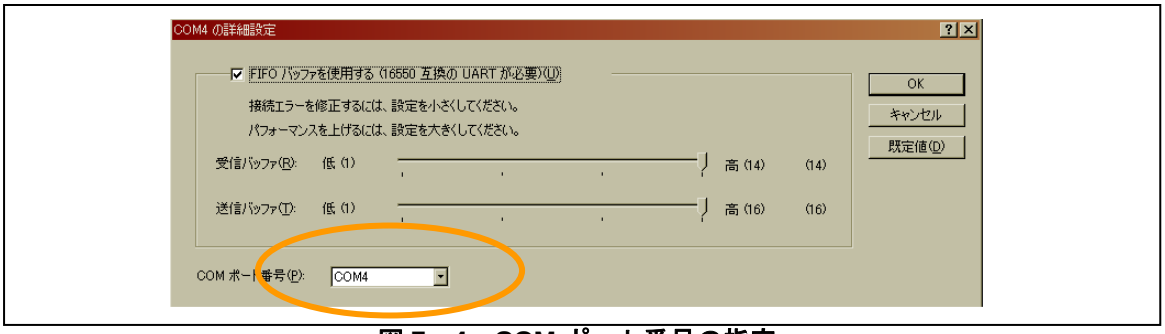


図 7−4　COM ポート番号の指定

備考 1.ほかのデバイスで使用するポート番号と重ならないようにしてください。

備考 2.この変更後はすぐに新しいポート番号が有効になりますが，デバイスマネージャの一覧表示にはすぐに反映されないことがあります。

(2) プロパティの変更

Windows のデバイスマネージャで使用されるデバイスの属性など一部の情報は，任意のものに変更できます。変更可能な部分を次に示します。

<1>　デバイス名（デバイス一覧）

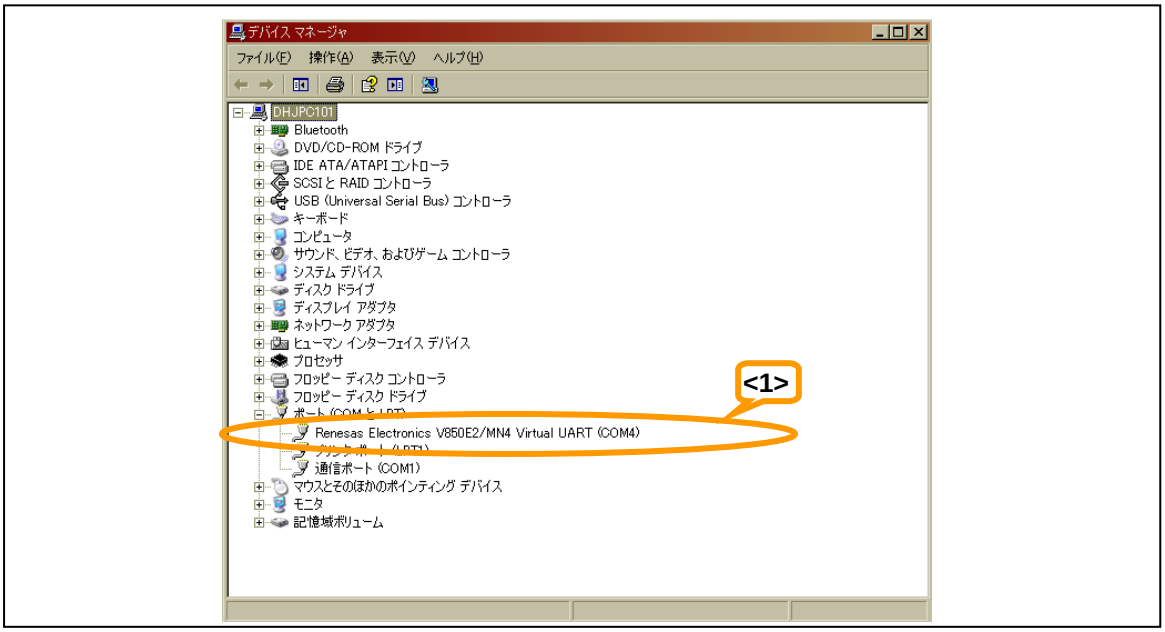


図 7−5　デバイスマネージャ上のデバイス名

<2>　デバイス名，製造元名，バージョン（デバイスのプロパティ）

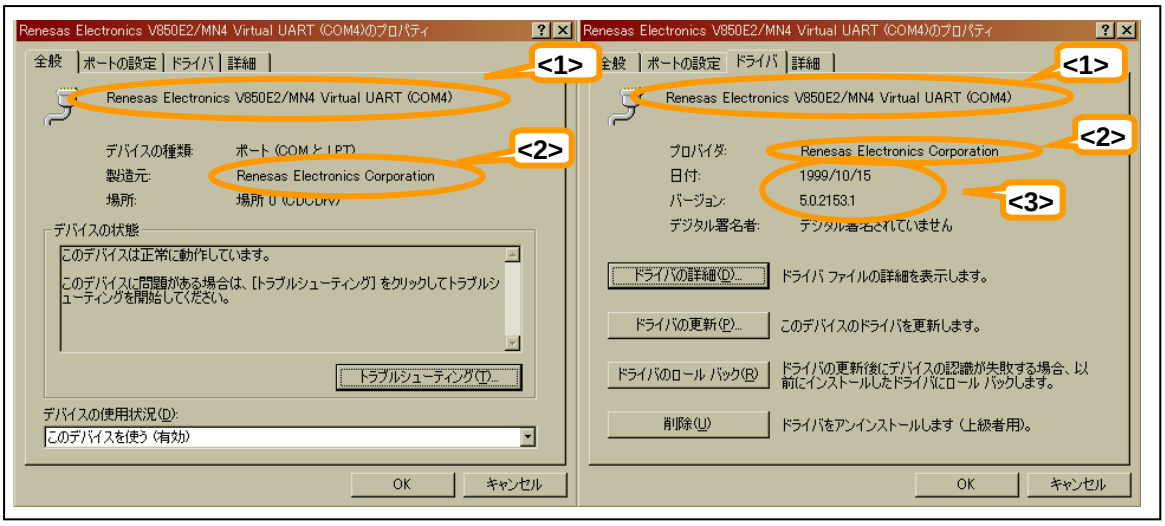


図 7−6　デバイスのプロパティ

これらは，ホスト・ドライバ（INF ファイル）に記述されている情報を元に表示されるため，INF ファイルを書き換えることで変更できます。INF ファイル内で前述の例の番号に対応する部分は次のとおりです。

```
1   ; .inf file (Win2000,XP):
2   [Version]
3   Signature="$Windows NT$"
4   Class=Ports
5   ClassGuid={4D36E978-E325-11CE-BFC1-08002BE10318}
6
7   Provider=%RENESAS%
8   LayoutFile=layout.inf
9   DriverVer=10/15/1999,5.0.2153.1                                          <3>
10
11  [Manufacturer]
12  %RENESAS%=RENESAS
13
14  [RENESAS]
15  %RENESASV850E2MN4%=Reader, USB¥VID_045B&PID_0200
16
17  [Reader_Install.NTx86]
18  ;Windows2000
19
20  [DestinationDirs]
21  DefaultDestDir=12
22  Reader.NT.Copy=12
23
24  [Reader.NT]
25  CopyFiles=Reader.NT.Copy
26  AddReg=Reader.NT.AddReg
27
28  [Reader.NT.Copy]
29  usbser.sys
30
31  [Reader.NT.AddReg]
32  HKR,,DevLoader,,*ntkern
33  HKR,,NTMPDriver,,usbser.sys
34  HKR,,EnumPropPages32,,"MsPorts.dll,SerialPortPropPageProvider"
35
36  [Reader.NT.Services]
37  AddService = usbser, 0x00000002, Service_Inst
38
39  [Service_Inst]
40  DisplayName = %Serial.SvcDesc%
41  ServiceType = 1 ; SERVICE_KERNEL_DRIVER
42  StartType = 3 ; SERVICE_DEMAND_START
43  ErrorControl = 1 ; SERVICE_ERROR_NORMAL
44  ServiceBinary = %12%¥usbser.sys
45  LoadOrderGroup = Base
46
47  [Strings]
48  RENESAS = "Renesas Electronics Corporation"                              <2>
49  RENESASV850E2MN4 = "Renesas Electronics V850E2/MN4 Virtual UART"         <1>
50  Serial.SvcDesc = "USB Serial emulation driver"
```

図 7−7　INF ファイル "MN4_CDC_XP.inf" の記述

(3) ベンダ ID とプロダクト ID の変更

デバイス・ディスクリプタ内のベンダ ID とプロダクト ID を変更した場合は，ホスト・ドライバ（INF ファイル）にも同じ内容を設定する必要があります。

INF ファイルでは，リスト 6−2 の 15 行目に次のような形式でベンダ ID とプロダクト ID を記述してください。

ベンダ ID　　：　"VID_" に続けて 4 桁の 16 進数で表記
プロダクト ID ：　"PID_" に続けて 4 桁の 16 進数で表記

7.3　関数の利用

使用頻度と汎用性の高い処理が定義済みの関数として用意されていますので，アプリケーションの記述を単純化でき，コード・サイズの節減にもつながります。各関数の詳細は 4. 3 関数の仕様を参照してください。
リストに示したサンプル・アプリケーションの次に示す部分は，定義済みの各種処理の応用例として再利用可能です。

(1) ユーザ・データ用 FIFO 状態確認

図 7-1 の 24 行目では, ユーザ･データ用 FIFO 状態通知処理関数(usbf850_get_bufinit_flg)を呼び出し、ユーザ･データ用 FIFO 初期化フラグ "usbf850_bufinit_flg" を監視しています。このフラグはサンプル・ドライバで独自に定義されているフラグで，サンプル・ドライバの INTUSFA0I1 割り込みで通知される Bus Reset 処理、および、クラス･リクエストの Set Line Coding リクエスト処理で FIFO の初期化が実行されるとセット(1)されます。サンプル･アプリケーションでは、この FIFO の初期化を契機にして、ユーザの送受信処理のエラー状態をクリア(0)します。

(2) ユーザ・データ受信処理

サンプル・ドライバでは，受信データの取り込みをデータサイズの取得とデータのコピーの 2 段階に分けて，それぞれの処理を定義した関数を用意しています。実際に受信したサイズを元に受信処理を実行する場合は、受信データサイズを確認することが出来ます。但し、1 回の受信処理で処理出来る最大のデータサイズは、1 パケットで受信されるデータサイズ以下である事に注意して下さい。サンプル･アプリケーションでは、バッファサイズが決まっている場合の使用例になっており、図 7-1 の 26 行目のユーザ・データ受信処理関数(usbf850_recv_buf)では、受信データがある場合、使用するエンドポイントから受信したデータを読み出します。

(3) ユーザ・データ送信処理

図 7-1 の 29 行目にあるユーザ・データ送信処理関数(usbf850_send_buf)では，送信データがある場合、使用するエンドポイントの FIFO の状態を確認し、FIFO Empty であればデータの書き込みを実行します。FIFO Full の場合、エラー終了します。また、送信データが無く、前に送信されたパケットのデータサイズが Max Packet Size に等しい場合、Null パケットの送信処理を実行します。これは、コミュニケーション・デバイス･クラスの仕様で、データの最終パケットが Max Packet Size に等しい場合、最終データである事をホスト側に通知する目的で Null パケットを送信する事になっている為です。

8. スタータ・キット概説

この章では，(株)マイダス・ラボ社製の V850E2/MN4 向けスタータ・キット RTE-V850E2/MN4-EB-S について説明します。

8.1 スタータ・キット概要

RTE-V850E2/MN4-EB-S は, V850E2/MN4 を使用したアプリケーション･システムの開発を体験できるキットです。ホスト・マシンに開発ツールや USB ドライバをインストールしてこのキットを MINICUBE 接続するだけで, プログラム作成からビルド, デバッグ, 動作確認といった一連の開発フローに対応できます。

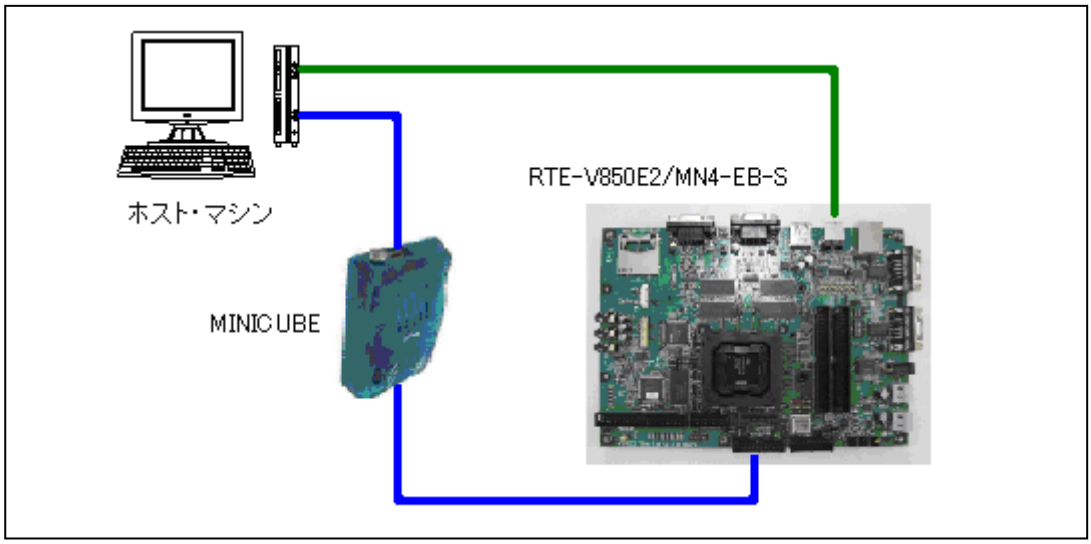


図 8－1　RTE-V850E2/MN4-EB-S の接続イメージ

8.2 スタータ・キット特徴

RTE-V850E2/MN4-EB-S には次のような特徴があります。

- 2 系統のメモリ・コントローラ, DMA，タイマ・アレイ,UART, CSI , CAN, A/D コンバータ, USB ファンクション・コントローラ,USB ホスト・コントローラ，イーサネット・コントローラなどの周辺機能を内蔵
- 入力 7 本，入出力 181 本の I/O ポートを装備
- 統合開発環境（CubeSuite/Multi/IAR Embedded Workbench）と組み合わせて効率的な開発を実現

8.3 主な仕様

RTE-V850E2/MN4-EB-S の主な仕様は次のとおりです。

○CPU	μPD70F3512（V850E2/MN4）
○動作周波数	200 MHz(PLL による 20 逓倍機能)
○インタフェース	USB コネクタ 2 基搭載（USB ホスト A タイプ×1 基，USB ファンクション B タイプ×1 基）
	N-Wire 用コネクタ
	UART2 基搭載
	CAN2 基搭載
	Ethernet コネクタ搭載
○対応機種	ホスト・マシン：USB インタフェース付き PC/AT 互換機
	OS：Windows 2000，Windows XP
○動作電圧	5.0 V
○本体寸法	W200×D150（mm）

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2010.06.30	—	初版発行
1.01	2011.04.28	全頁	ルネサスエレクトロニクスへの統合に伴うフォーマット改訂 6 章に GHS 版/IAR Embedded Workbench 版の内容を追加

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. 未使用端子の処理

   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。

   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。

2. 電源投入時の処置

   【注意】電源投入時は，製品の状態は不定です。

   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。

   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。

   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. リザーブアドレスのアクセス禁止

   【注意】リザーブアドレスのアクセスを禁止します。

   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

4. クロックについて

   【注意】リセット時は、クロックが安定した後、リセットを解除してください。

   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。

   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

5. 製品間の相違について

   【注意】型名の異なる製品に変更する場合は、事前に問題ないことをご確認下さい。

   同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry

Colophon 1.0